Characterizing Search Activities on Stack Overflow

Jiakun Liu
Zhejiang University
Hangzhou, Zhejiang, China
jkliu@zju.edu.cn

Sebastian Baltes
The University of Adelaide
Adelaide, Australia
sebastian.baltes@adelaide.edu.au

Christoph Treude
The University of Adelaide
Adelaide, Australia
christoph.treude@adelaide.edu.au

David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

Yun Zhang
Zhejiang University City College
Hangzhou, Zhejiang, China
yunzhang@zucc.edu.cn

Xin Xia*
Huawei
Hangzhou, Zhejiang, China
xin.xia@acm.org

ABSTRACT

To solve programming issues, developers commonly search on Stack Overflow to seek potential solutions. However, there is a gap between the knowledge developers are interested in and the knowledge they are able to retrieve using search engines. To help developers efficiently retrieve relevant knowledge on Stack Overflow, prior studies proposed several techniques to reformulate queries and generate summarized answers. However, few studies performed a large-scale analysis using real-world search logs. In this paper, we characterize how developers search on Stack Overflow using such logs. By doing so, we identify the challenges developers face when searching on Stack Overflow and seek opportunities for the platform and researchers to help developers efficiently retrieve knowledge.

To characterize search activities on Stack Overflow, we use search log data based on requests to Stack Overflow's web servers. We find that the most common search activity is reformulating the immediately preceding queries. Related work looked into query reformulations when using generic search engines and found 13 types of query reformulation strategies. Compared to their results, we observe that 71.78% of the reformulations can be fitted into those reformulation strategies. In terms of how queries are structured, 17.41% of the search sessions only search for fragments of source code artifacts (e.g., class and method names) without specifying the names of programming languages, libraries, or frameworks. Based on our findings, we provide actionable suggestions for Stack Overflow moderators and outline directions for future research. For example, we encourage Stack Overflow to set up a database that includes the relations between all computer programming terminologies shared on Stack Overflow, e.g., method name, data structure name, design pattern, and IDE name. By doing so, Stack Overflow could improve the performance of search engines by considering related programming terminologies at different levels of granularity.

*Corresponding author

ESEC/FSE '21, August 23–28, 2021, Athens, Greece

ACM ISBN 978-1-4503-8562-6/21/08.
https://doi.org/10.1145/3468264.3468582

CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**.

KEYWORDS

Stack Overflow, Data Mining, Query Reformulation, Query Logs

ACM Reference Format:
Jiakun Liu, Sebastian Baltes, Christoph Treude, David Lo, Yun Zhang, and Xin Xia. 2021. Characterizing Search Activities on Stack Overflow. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3468264.3468582

1 INTRODUCTION

Stack Overflow is a popular and active knowledge base that hosts questions and answers related to programming technologies [11–13]. When developers seek solutions to programming issues, Stack Overflow encourages them to "search and research" before posting questions [4]. Developers commonly search on Stack Overflow [57] to leverage existing knowledge. For example, if developers could find the same problems and corresponding answers as the programming issue they are experiencing, they do not need to wait for a response to their questions. Moreover, the knowledge related to the same programming issue would not be scattered across threads [14].

However, there is a gap between the information need of developers and the results retrieved by search engines. Although developers may have a specific information need in mind, they might not understand how to express their needs clearly in a natural language query. Developers may need to search multiple times on Stack Overflow to find the information that they need. To help developers efficiently retrieve knowledge on Stack Overflow, prior studies proposed techniques to reformulate queries [23, 59] and generate summarized answers [58]. However, few studies have performed a large-scale analysis using search logs to characterize how developers search on Stack Overflow. By doing so, we could identify the challenges developers faced when searching on Stack Overflow and seek opportunities for Stack Overflow and researchers to help developers efficiently retrieve knowledge.

To characterize how developers search on Stack Overflow, we used search log data collected from requests to Stack Overflow's web servers. Stack Overflow shared the search log data with us

under an NDA agreement. This shared data records the requests to Stack Overflow's web servers from December 1, 2017, to November 31, 2018. We extract the search sessions that are complete linear navigation sequences with at least one search request. In complete linear navigation sequences, the referrer web page of one request is the target web page of the previous request. We do not consider sequences that have more than one successor to a predecessor, or sequences with gaps, because such sequences do not contain a clear order of reformulation events. To describe how developers search on Stack Overflow, we **consider the page visited before a certain search event was triggered on Stack Overflow.** For example, searching while having a certain post open and searching from the result page of a previous search request are different search activities because they have different preceding pages. Overall, we considered 14,126,647 search activities on Stack Overflow in 4,981,786 search sessions. We structure our study by answering the following research questions:

(1) **What are the search activities on Stack Overflow?**
The most common search activity is reformulating the immediately preceding query. When developers search from a post, the time interval between the request to the post and the request related to a new search is the largest compared to other activities.

(2) **What are the differences between searching on Stack Overflow and general search in terms of query reformulation?**
71.78% of the reformulations on Stack Overflow can be assigned to the reformulation strategies identified for general search as reported by Huang and Efthimiadis [33]. For the reformulations that cannot be assigned to the strategies identified in general search, only 26.69% of them have domain-specific characteristics, e.g., substituting a name of a function defined for a programming language with another name of a function with similar functionality.

(3) **What is the nature of queries on Stack Overflow?**
17.41% of the search sessions only search for fragments of source code artifacts (e.g., class and method names) without specifying the names of programming languages, libraries, and frameworks. Developers use queries with different semantic tags for different types of search intentions. They most frequently search for manipulating data type or function of a certain programming languages and their corresponding technologies, e.g., library and framework.

Based on our findings, we provide actionable suggestions for Stack Overflow moderators and researchers. For example, we suggest that researchers could propose specific optimization approaches for the reformulations that have domain-specific characteristics. We encourage Stack Overflow to set up a database similar to WordNet that includes the relations between all programming terminologies shared on Stack Overflow, e.g., method name, data structure name, and design pattern name. By doing so, Stack Overflow could return more relevant search results by considering programming terminologies that are related in terms of their technical aspects, and developers would not need to reformulate their queries by trying programming terminologies at different levels of granularity, e.g., replacing a method name with a class name or library name.

2 RELATED WORK

2.1 Analyzing Search Logs

Search engine logs (i.e., search logs) record the HTTP requests to search engine web servers [24]. Many works use search engine logs to characterize how users refine their queries as well as how users click on suggested results.

To model the user behaviors when searching, prior studies extracted click data from search logs [20, 25, 36, 40]. For example, Radlinski et al. used the time to first click on the search result page to reflect the quality of the result presentation [46]. Kim et al. considered the server-side click dwell time as a sum of the actual click dwell time and the time to formulate the next query [36].

To better understand the search intentions, prior studies extract queries from search logs. Various classifications and taxonomies have been proposed to understand users' search queries from different viewpoints, including intent taxonomy, topic taxonomy, and performance taxonomy [21, 39, 49, 57]. For example, the KDD Cup 2005 provides a formal two-level taxonomy, with 67-second level nodes and 800,000 Internet user search queries [39].

Different from the documents with well-formed sentences that can be used for Information Retrieval (IR), the keyword-based queries are usually short and lack sentence structures. Part-Of-Speech (POS) tags and linguistic structures carry meaningful information to match queries and documents. Manshadi et al. investigated how to assign a label from a set of pre-defined domain-specific semantic labels to every word in the query [41]. Hearst et al. identified a set of lexico-syntactic patterns to recognize hyponyms from free-form text [31]. This approach helps ground various studies on extracting semantic phrases [28, 45].

Considering that web documents and user queries are created by different sets of users, they may use different vocabularies and distinct language styles. There exists a "lexical gap" between web documents and user queries. Users have to reformulate queries to find related information in search engines. To better understand how users reformulate their queries, taxonomies of query reformulation have been developed for different types of search [18, 22, 54, 56]. For example, Teevan et al. constructed a taxonomy by looking through query logs, and implemented algorithms to detect a subset of the reformulation strategies [54].

Different from these studies, to the best of our knowledge, this is the first large-scale study of search activities on a domain-specific large Q&A website. We characterize different search activities on Stack Overflow, identify the reformulation strategies adopted by developers when searching, and investigate the contents of queries.

2.2 Analyzing Search Activities in Software Engineering

Prior studies characterize how developers search for programming-related knowledge during the development process [44, 50, 57]. For example, Singer et al. used a combination of surveys, interviews, observations, and collecting tool usage statistics to characterize the search activities during the development process [50]. They found that software developers spend most time in searching for information as well as reading documentation and source code. Xia et al. collected search queries from 60 developers and surveyed 235 software engineers to characterize what developers frequently searched for [57]. They observed that searching for explanations for unknown terminologies is the most frequent search task. Different from prior studies, our work uses the search logs of the popular question and answering platform to perform a large-scale analysis of the search activities.

Bajracharya and Lopes analyzed the usage log of a code search engine (i.e., Koders) [19]. They observed that users who find code search engines usable already know with high level of specificity what to look for. Sadowski et al. used a combination of survey and log-analysis methodologies to characterize how developers search for code on Google [48]. They observed that programmers are looking for code with which they are somewhat familiar. Different from prior studies, our work analyzes the search logs in the popular question and answering platform (i.e., Stack Overflow).

Besides, a number of studies focused on answer-seeking on Stack Overflow [43, 58, 59]. For example, Zhang et al. proposed an interactive query refinement approach for question retrieval to help users recognize and clarify technical details missed in queries [59]. However, it is still unclear what are the similarities and differences between the query reformulation processes in searching on Stack Overflow and general search. Kaibo et al. proposed an automated software-specific query reformulation approach based on deep learning [23]. They performed a formative study on the users' activity logs from Stack Overflow. Different from prior studies, our work compares the query reformulation process in Stack Overflow with that in general search. We identify the types of reformulations that are distinct in Stack Overflow.

3 STUDY SETUP

Our raw dataset contains all internal requests processed by Stack Overflow's web servers, including the timestamps and other metadata such as the referrer, target URL, and anonymized user identifier. Internal means that requests of users coming from external websites were excluded, yielding 747,421,780 requests in total, ranging from December 1, 2017, to November 31, 2018. The anonymized user identifiers were created for logged-in users, users tracked via cookies, or generated for sessions based on available information such IP addresses. We had no information on how users were identified, we just had access to the randomly generated identifiers.

To filter out search logs from raw dataset, we considered **search sessions** as **complete linear navigation sequences with at least one search request**. In complete linear navigation sequences, the referrer web page of one request is the target web page of the previous request. We excluded sequences having more than one potential successor for a target event, i.e. non-linear sequences, and sequences with gaps, i.e. incomplete sequences. For example, a user may trigger multiple search requests from different browser tabs with the same Stack Overflow page open, while being tracked as described above. In that case, we cannot linearly order the user's navigation events. Further, since we only had access to internal requests, a sequence may contain gaps if a user navigated to external websites and later came back to Stack Overflow. We excluded such sequences, because users might have triggered other search requests with general search engines during the gaps, giving us an incomplete view of search activities. Hence, we focused on complete linear sequences with a total order over the navigation events.

To extract such complete linear navigation sequences, we first grouped all requests based on user identifiers and ordered them chronologically. We distinguished different sessions if the time interval between consecutive requests was more than six minutes apart, following Sadowski et al.'s approach [23]. As a result, 747,421,780 web server requests were grouped into 98,495,404 navigation sequences. Secondly, we excluded 74,613,186 (75.75%) non-linear sequences and incomplete sequences. Thirdly, we utilized heuristics based on the timestamps and request targets to filter out bot traffic. This ended up with 18,269,793 (18.55%) linear non-bot sequences. We further removed page refresh events. This ended up with 16,164,506 sequences (88.48% of the linear non-bot sequences). Of those sequences, 4,981,786 (30.8%) had at least one search event. All those steps were developed in an iterative process, involving qualitative analysis of samples of sequences to detect problematic instances (non-linear sequences, bot traffic, page refreshes, etc.). [1] For the sessions with more than one search request, we consider the later queries to be reformulations of the earlier ones rather than queries requesting new information. Section 5.2 discusses the threats to validity related to the time interval.

In the dataset, **each request was assigned to certain event types**, determined based on the requested resource. For example, the event type `Search` is related to triggering a search using Stack Overflow's search box. The event type `Post` is related to requesting a certain post, i.e. question or answer, e.g. by clicking on a link in the search result list. The event type `Home` is related to requesting the home page of Stack Overflow. We use the artificial event type `Begin` to describe the beginning of a search session and the artificial event type `End` to describe the end of a search session. Note that developers could search from any search result page, post page, or the home page of Stack Overflow using the search box in the header of the web page. Developers could visit the home page by clicking the Stack Overflow icon in the header of the web page from the search result page or the post page. Developers also could visit a post from the search result page or the home page. To describe how developers search on Stack Overflow, **we consider the page from which search events were triggered when we refer to search activity on Stack Overflow (i.e., we focus on consecutive pairs of events).** For example, searching from a post and searching from a search result page are different search activities because they have different preceding events. As a result, we obtain search log data with 14,126,647 search activities (i.e., consecutive pairs of events) in 4,981,786 search sessions.

4 RESULTS

4.1 RQ 1: What Are the Search Activities on Stack Overflow?

Motivation: Prior studies identified that developers commonly search during their development process [50, 57]. However, few studies performed a large-scale analysis to characterize how developers search on Stack Overflow. Here, we would like to characterize the search activities (i.e., consecutive pairs of events) using the search log data. More specifically, we would like to understand which web pages are frequently requested by developers when browsing a certain web page, the beginning and the end of search sessions, and the time interval between the requests related to the consecutive pairs of events in different search activities. By doing so, we could identify opportunities and challenges for researchers and Stack Overflow to help developers search on Stack Overflow.

[1] The code is available on https://zenodo.org/record/4730525.

Approach: To calculate the occurrences of different search activities on Stack Overflow, we count the number of different types of consecutive pairs of events.

Then, we investigate whether the occurrence of certain search activity is related to the occurrence of another search activity. More specifically, we would like to mine the patterns of the co-occurrences of different types of consecutive pairs of events. We use the Apriori algorithm to analyze the association rules between different search activities (i.e, consecutive pairs of events) [16, 17]. We measure the performance of rules with the support and confidence metrics. Support expresses the ratio in which the rule can be found in the data set. Confidence expresses the ratio in which the rule is correct.

We calculate the time interval between the request to the preceding page and the request to the target page for each search activity. We compare the differences between different types of search activity in terms of the time interval of the requests.

Results: The most common search activity is *reformulating the immediately preceding query* (i.e., Search→Search). We identify 336 types of search activity (i.e., consecutive pairs of events). The top 9 types of search activities account for 86.07% of the search activities on Stack Overflow. We refer to the top 9 types of search activities as the dominant search activities. Table 1 presents the dominant search activity types on Stack Overflow. 17.20% of the search activities are from a search result page to another search result page. Following that, ***visiting a post from search result page* (i.e., Search→Post) is the second most common search activity**. 13.86% of the search activities are from a search result page to a post. This indicates that developers need to frequently identify whether the post excerpts in the search result list are related to their search intentions.

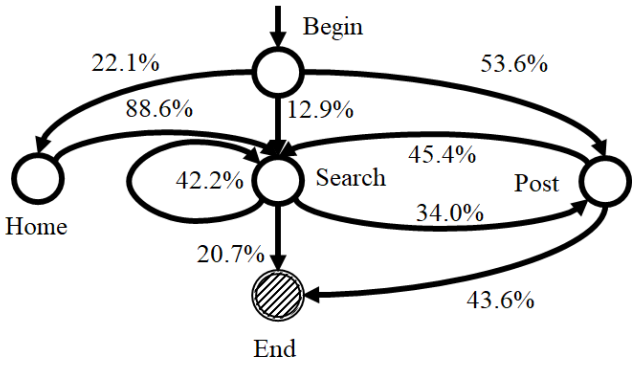

Figure 1: Dominant types of search activity. We present the proportion of different types of target pages when developers browse different types of preceding pages.

Next, we would like to discuss the beginning of search sessions. Figure 1 depicts the dominant types of search activity on Stack Overflow.[2] Table 2 shows the rules of the top 10 most frequently co-occurring search activity types related to the beginning and the end of the search sessions (i.e., the beginning of the search session is in left-hand side, or the end of the search session is in right-hand side). We focus on the rules with support > 0.129 because the proportion of the search sessions with the 9-th most common search activity (i.e., Begin→Search in Table 1) is 12.9%. Table 2 shows that *searching from the posts* (i.e., Post→Search) is frequently co-occurring with the beginning of the search session. One possible reason is that developers cannot obtain desired knowledge from the web pages included in the search result list in external search engines. Developers continue their search tasks using the search engine provided by Stack Overflow.

Next, we would like to discuss the end of search sessions. Table 2 shows that *visiting a post from the search result page* is frequently co-occurring with the end of the search session. **62.73% (i.e., 3,125,053) of search sessions end up in posts.** At the end of the search session, developers may find posts related to their search intentions. They click the posts and finish the search task. However, developers may not find interesting posts related to their search intentions. Table 2 shows that *reformulating the immediately preceding query* is frequently co-occurring with the end of the search session. They give up reformulating queries and do not end the search session in posts. **For the sessions that do not end up in posts, only 0.01% of them request the QuestionAsk page, i.e., propose questions on Stack Overflow.** This shows that developers commonly do not ask questions when they cannot find the knowledge related to their programming problems.

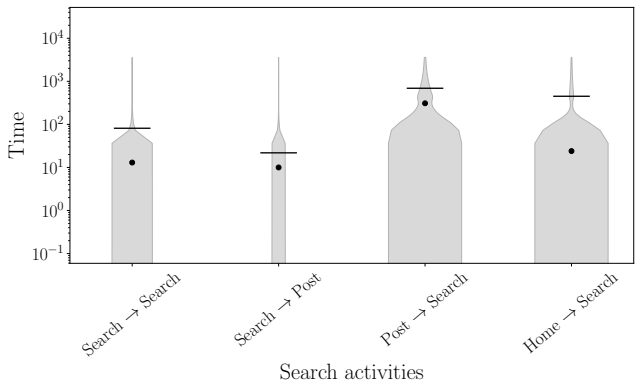

Figure 2: Time interval between the request to preceding page and the request to the target page in dominant types of search activity. The y-axis is in log scale.

Figure 2 shows the distribution of the time interval between the request to the preceding page and the request to the target page in dominant types of search activity. To check whether the differences in the time interval are statistically significant between different types of search activity, we perform the Kruskal-Wallis H-test [37]. The Kruskal-Wallis H test is a non-parametric test for comparing whether two or more independent samples originate from the same distribution. The null hypothesis is that there is no difference in the time interval between different types of search activity. As a result, the differences in the time interval between different types of search activity are significant (p-value < 0.05). One possible reason is that developers need different amounts of time in different types of search activity. To identify which types of search

[2]We do not present the proportion of the target page types that are not dominant search activity types.

Table 1: Top 9 most common search activity types on Stack Overflow. We present the types of search activities, the corresponding frequencies, the proportion among all search activities, the examples, and the corresponding description.

Activity	#	%	Predecessor	Target	Description
Search→Search	4,143,291	17.20%	/search?q=java	/search?q=java+linux	reformulate the immediately preceding query
Search→Post	3,338,051	13.86%	/search?q=java	/q/215497	visit a post from search result page
Post→Search	2,817,653	11.70%	/q/215497	/search?q=java	search from the posts
Post→End	2,701,072	11.21%	/q/215497		the search session ends up in post
Begin→Post	2,670,252	11.08%		/q/215497	the search session begins with post
Search→End	2,036,898	8.46%	/search?q=java		the search session ends up with search result page
Home→Search	1,285,261	5.34%	/home	/search?q=java	search from the home page
Begin→Home	1,100,320	5.57%		/home	the search session begins from the home page
Begin→Search	642,797	3.67%		/search?q=java	the search session begins from the search page

Table 2: Top 10 frequently co-occurring search activity types related to the beginning and the end of the search sessions.

left-hand-side	right-hand-side	support
Search→Post	Post→End	0.536
Begin→Post	Post→Search	0.488
Begin→Post	Search→Post	0.365
Begin→Post,Search→Post	Post→Search	0.348
Begin→Post,Post→Search	Search→Post	0.348
Begin→Post	Post→Search, Search→Post	0.348
Search→Search	Search→End	0.340
Post→Search	Post→End	0.324
Post→Search,Search→Post	Post→End	0.322
Search→Post	Post→End, Post→Search	0.322

Table 3: P-values and Cliff's delta of the differences between different types of search activity in terms of time interval.

	Search→Post	Post→Search	Home→Search
Search→Search	+ small	- large	- small
Search→Post		- large	- medium
Post→Search			+ medium

activity are associated with a larger time interval, we perform a Dunn's test with Bonferroni correction to determine which groups differ from each other [27]. Dunn's test can be used for the post-hoc analysis for the specific sample pairs. Bonferroni correction is used to counteract the problem of multiple comparisons [15]. We also calculate Cliff's delta to measure the effect size [26]. Cliff's delta is a non-parametric effect size measure that can evaluate the amount of difference between two groups. Romano et al. define an absolute delta of less than 0.147, between 0.147 and 0.33, between 0.33 and 0.474 and above 0.474 as "Negligible", "Small", "Medium", "Large" effect size, respectively [47]. Table 3 shows the p-values and Cliff's deltas.

The time interval between requests when developers *visit a post from the search result page* (i.e., Search→Post), is the smallest compared with other search activities, followed by when developers *reformulate the immediately preceding query* (i.e., Search→Search). One possible reason is that developers spend little time in understanding the difference and similarities between the post excerpts presented in the search result list page and their search intention.

When developers *search from a post* (i.e., Post→Search), the time interval between requests is the largest. One possible reason is that it takes the most time for developers to come up with the next query after browsing the posts on Stack Overflow. We suggest that researchers could design a tool for developers to label the sentences in posts that are the most similar to and the most different from their search intention to better reformulate queries.

When developers *search from the home page* (i.e., Home →Search), the time interval between requests is the second largest. One possible reason is that it takes a long time for developers to write the initial query on Stack Overflow. Developers may not be aware of how to write the initial query to express their search intention. This motivates us to analyze how developers express their search intention into queries in Section 4.3.

> The most common search activity is reformulating the immediately preceding query. For the sessions that do not end up in posts, only 0.01% of them ask questions on Stack Overflow. When developers search from a post, the time interval between requests is the largest. This is followed by searching from the home page.

4.2 RQ 2: What Are the Differences Between Searching on Stack Overflow and General Search in Terms of Query Reformulation?

Motivation: Prior studies have developed tools to help developers reformulate their queries on Stack Overflow [43, 59]. However, they did not compare searching on Stack Overflow with general search in terms of the query reformulation process. By doing so, researchers could propose specific optimization approaches for the types of reformulations that are distinct on Stack Overflow.

Approach: To identify the reformulation strategies in the search activities in the field of software engineering, we follow Huang and Efthimiadis's work [33]. Huang and Efthimiadis learned from the search logs of general search engines and created the taxonomy of reformulation strategies. We use the taxonomy of Huang and

Efthimiadis's work as a starting point because they summarized the reformulation strategies identified in prior studies [30, 35, 38, 54, 56]. This taxonomy is still popular in the field of query reformulation [32, 51–53]. Besides, Huang and Efthimiadis proposed a heuristic algorithm to determine the taxonomy reformulation strategies [8]. We use this heuristic algorithm to determine the taxonomy of reformulation strategies [33]. As a result, out of 4,048,518 reformulations on Stack Overflow, 2,905,998 of them can be fitted into the reformulation strategies that are identified in Huang and Efthimiadis's work (i.e., determined reformulations). 1,142,520 of the reformulations cannot be fitted into the taxonomy of reformulation strategies using the heuristic algorithm (i.e., undetermined reformulations).

Huang and Efthimiadis characterized the reformulations that cannot be fitted into the taxonomy of reformulation strategies using the heuristic algorithm [33]. Huang and Efthimiadis observed there are three general types of undetermined reformulations, i.e., rephrasing semantically with sophisticated semantic associations, reformulations with more than one reformulation strategy, and classifier rule limitations (e.g., spelling correction with three or more character edits, word substitutions with words absent from the WordNet database, URL stripping with second-level domains which are not stripped, and abbreviations not being a substring prefix). Note that adding words and removing words together were not considered a multi-reformulation since any query can be transformed into any other query. Following their work, we randomly sampled a statistically representative sample of 386 undetermined reformulations, using a 95% confidence level with a 4.99% confidence interval. Then we manually performed an open coding process to identify the strategies of undetermined reformulations. This process involves 3 phases and is performed by two authors (A1 and A5) of this paper:
Phase I: We randomly selected 50 undetermined reformulations from the sampled 386 undetermined reformulations. The first two authors used the coding schema provided by Huang and Efthimiadis's work to categorize the selected 50 undetermined reformulations collaboratively [33]. During this phase, the coding schema of the types of undetermined reformulations on Stack Overflow was revised and refined. We performed the refinement because: (1) A large number of undetermined reformulations are related to adding or removing bug information. (2) A large number of undetermined reformulations are related to replacing programming terminologies. (3) There are undetermined reformulations related to translating queries from one natural language to English. (4) There are undetermined reformulations where the search intentions are expressed in different queries. For example, *[apache-spark] evaluation* is reformulated to *[apache-spark] ml* at first. Then *[apache-spark] ml* is reformulated to *[apache-spark] ml evaluation*. We have to read all queries in a session to identify developers' search intentions.
Phase II: The first two authors applied the resulting coding schema of Phase I to categorize the remaining 334 undetermined reformulations independently. They were instructed to take notes regarding the deficiency and ambiguity of the coding schema for categorizing certain undetermined reformulations. The inter-rater agreement (Cohen's kappa) of this stage is 0.69, indicating that the agreement level is substantial [55].
Phase III: The first two authors discussed the coding results obtained in Phase 2 to resolve the disagreements. We also invited another Ph.D. student, who is not an author of the paper, to discuss the disagreements. For example, for the undetermined reformulation *[geoserver] timestamp →[geoserver] csw*, A1 considered this reformulation as searching for different topics because *csw* is not related to *timestamp* in Wikipedia [3]. A5 considered this reformulation as technical entity substitution because CSW is a service related to time in geoserver [1]. We finally considered this reformulation as a technical entity substitution. We did not identify new categories in Stage 3. At the end of Stage 3, we obtained the final coding schema and the final coding results of the sampled 386 undetermined reformulations.
Results: 71.78% of the reformulations can be fitted into the reformulation strategies identified in general search, e.g., users search non-technical information on Google. Table 4 shows the categories of reformulation strategies using the algorithms proposed for general search. Table 5 shows the categories of reformulation strategies for the undetermined reformulations. Prior studies propose domain-specific tools to help developers to search on Stack Overflow [23]. However, they did not characterize the similarities and differences between the general search and searching for programming knowledge on Stack Overflow. Table 5 shows that 67.88% of the undetermined reformulations can be observed in Huang and Efthimiadis's work [33]. We suggest researchers could focus on the reformulations that cannot be detected in general search engines.
4.2.1 Strategies of Determined Reformulations. **Developers specialize the search intention the most during query reformulations.** More specifically, we observe that 22.99% of the query reformulations are related to adding words to previous queries. Prior studies have identified that developers commonly specialize their search intention by adding words to the first queries [34]. This shows that for the search sessions with query reformulations, developers commonly search with queries with limited information at the beginning. Developers add more details related to their search intention during the reformulation process. Following that, **performing another search with advanced search mechanism is the second most common reformulation strategy**. We observe 17% of the reformulations with the same queries but different parameters in the query string of the URL, e.g., `tab=relevance`, and `tab=newest`. For example, developers can order the search results by time or relevance [5, 9].

Generalizing search intention is the third most common reformulation strategy. More specifically, we observe that 15.81% of the query reformulations are related to removing words from previous queries. Similarly, prior studies have identified that developers commonly generalize their search intention by removing words from the first queries [34]. One possible reason is that the queries have too much information. The Stack Overflow search engine has poor performance for complex queries, as developers have to hide part of their search intention with more general terminology. The last query in a search session may not be the query with the most search intention. This motivates us to read all queries in a session to understand the search intention.
4.2.2 Strategies of Undetermined Reformulations. Here, we would like to discuss the reformulations that cannot be fitted into the strategies identified in general search. We observe that **26.69% of the undetermined reformulations are related to the domain-specific**

Table 4: Categories of reformulation strategies. We also present the description of different types of reformulation strategies, the example query reformulation, and the corresponding proportion among all 4,048,519 reformulations.

Category	Description	Example		%
		previous	after	
add words	one or more words are added to the first query	python removing LS	python removing LS **control character**	22.99%
same	two queries are the same	FireMonkey	**tab=newest&q=**FireMonkey	16.83%
remove words	any number of words is removed from the first query resulting in the same words in both queries.	**[web-scraping]** mozenda date	mozenda date	15.81%
spelling correction	the Levenshtein distance is 2 or less	xlwings **mege**	xlwings **merge**	5.93%
abbreviation	corresponding words from the first and second queries are prefixes of each other	drf Unsupported media **typ**	drf Unsupported media **type**	3.25%
whitespace punctuation	only whitespace and punctuation are altered.	excel like **gridreactjs**	excel like **grid reactjs**	1.97%
word substitution	one or more words in the first query are substituted with semantically related words, determined from the Wordnet database	iOS photo save **direction**	iOS photo save **Orientation**	1.57%
substring	the second query is a strict prefix or suffix of the first query	**GLES20Canvas.nGetDisplayList**	**GetDisplayList**	1.24%
stemming	changing the word stems in the first query	[salesforce] D43 create test **account**	[salesforce] D43 create test **accounts**	0.81%
word reorder	the words in the initial query are reordered but unchanged otherwise.	**django pyodbc oracle**	**oracle pyodbc django**	0.68%
superstring	the second query contains the first query as a prefix or suffix	Access denied finding propert	Access denied finding property **"net.dns1"**	0.63%
url strip	the first and second queries are the same after removing ".com", "www.", and "http" from both sides.	bit-z.com	**www**.bit-z.com	0.03%
expand acronym	the first query is an acronym and the reformulation is a query consisting of the words that form the acronym.	cfd	**c**umulative **f**low **d**iagram	0.02%
form acronym	the second query is an acronym formed from the first query's words.	**a**mazon **p**ublisher **s**ervice	aps	0.01%

Table 5: Categories of reformulation strategies for the undetermined reformulations. We also present the description of reformulation strategies, the example, and the corresponding proportion among the sampled 386 reformulations.

Category	Description	Example		%
		previous	after	
Semantic rephrasing	Reformulation related to sophisticated semantic association.	opinion mining	sentiment analysis	23.06%
New topic	The queries search for different topics.	activesheet.range	select all cells	22.54%
Multiple reformulations	One reformulation with more than one reformulation strategy.	hwo to use excel exponentiation	excel Exponentiation	20.98%
Technical entity substitution	One or more technical terms in the first query are substituted with technical related words that are not identified in Wordnet database.	iOS alarm app tutorials	swift alarm app tutorials	18.91%
Add bug information	Add the description of the unexpected code behavior.	static inter view to layout	ClassNotFoundException when add view	4.15%
Remove bug information	Remove the description of the unexpected code behavior.	css not working in bootstrap mobile	override bootstrap css	3.63%
Compensation	Intention are scattered in more than one queries.	[apache-spark] evaluation	[apache-spark] ml	3.37%
Translation	The queries translate the first query into English.	création de dictionnaire python	dictionary python	2.59%
Other rule limitation	Insufficiently matched by a classifier rule.	https://www.Google maps.com/m.&chad.deep space& mars	https://www.Google maps.com/.deep space& mars	0.78%

characteristics. More specifically, 18.91% of the undetermined reformulations are related to the programming terminologies in previous queries being replaced by other programming terminologies. 4.15% and 3.63% undetermined reformulations are related to the developers directly introducing or removing a bug trace. Similar to the semantic relations in the WordNet database, technical relations can be similar functions (i.e., the functions that can achieve the same task), inheritance, and generalization, etc [60]. For example, a query *Promise.all* is reformulated to *async/await*, where *Promise.all* can be used to await multiple requests to finish with async/await syntax. We suggest researchers could set up a technical relation database similar to WordNet. The nodes in the database can be programming terminologies at different levels of granularity. The

relations in the database can record similar functions, control flow, or data flow.

71.78% of the reformulations can be fitted into the reformulation strategies identified in general search. The most common reformulation strategies are specializing the search intention, using advanced search mechanisms, and generalizing the search intention. For the reformulations that cannot be fitted into the strategies identified in the general search, 26.69% of them are related to the domain-specific characteristics.

4.3 RQ 3: What Is the Nature of Queries on Stack Overflow?

Motivation: Prior studies identified the tasks that are commonly searched during the development process across different platforms [57]. However, the intention when searching on Stack Overflow and how developers express their search intentions is still unclear. More specifically, we would like to characterize the query contents, the search intentions, and the relations between them. By doing so, we could provide suggestions for Stack Overflow in improving the performance of the search engine. Stack Overflow also could propose guidelines to help developers better express their search intention into queries.

Approach: To characterize query contents, we randomly sampled a statistically representative sample of 1,086 search sessions, using a 95% confidence level with a 3% confidence interval. Then we manually performed an open coding process to identify the semantic tags of the phrases in queries. This process involves 3 phases and is performed by three authors (A1, A5, and A6) of this paper:

Phase I: A1 first developed a draft coding schema of the categories of the semantic tags of the phrases in queries, using 100 sessions selected from the sampled 1,086 sessions. To identify the semantic tags of the phrase in queries, we first performed the query segmentation task, i.e., dividing each query into valid meaningful minimal segments. Following prior studies, we define a query q as a sequence $(w_1, \ldots, w_k)$ of k keywords [29]. A valid query segmentation determines for each pair w_i, w_{i+1} of consecutive keywords in q whether there should be a segment break between w_i and w_{i+1}. Hence, there are 2^{k-1} possible segment breaks for a k-keyword query and $k(k+1)/2$ potential segments. Note that Stack Overflow covers a wide range of knowledge across technologies. Then A1 searched all query segments in general search engines, e.g., Google and Bing. By doing so, A1 could identify the semantic meanings of the phrases in queries. Note that the semantic tags can be hierarchically organized. We obtain the **semantic hierarchy** of divided query segments when we search them in general search engines. For example, the *technology name*, `angular` is a *platform* for building web applications. *Platform* is the most fundamental semantic hierarchy for the terminology `angular`. For the divided query segments, we identify their semantic tags at the most fundamental semantic hierarchy level. Then A5 and A6 used the draft coding schema to categorize the semantic tags of the phrases in queries of 100 sessions collaboratively. During this phase, the draft coding schema of the semantic tags of the phrases in queries of the 100 sessions was revised and refined. At the end of Stage 1, we obtained 74 categories of semantic tags.

Phase II: A1 and A5 applied the resulting coding schema of Phase I to categorize the remaining 986 sessions independently. They were instructed to take notes regarding the deficiency and ambiguity of the coding schema for categorizing sessions. The inter-rater agreement (Cohen's kappa) for identifying semantic tags of this stage is 0.82, indicating that the agreement level is substantial [55].

Phase III: A1, A5, and A6 discussed the coding results obtained in Phase 2 to resolve the disagreements. At the end of Stage 3, we obtained the final coding schema and the final coding results of the sampled 1,086 sessions.

Then we perform a qualitative study to understand the search intentions of the queries in search sessions on Stack Overflow using the above samples. More specifically, we manually performed an open coding process to identify the categories of the search intentions on Stack Overflow. This process involves 3 phases and is performed by the three authors (A1, A5, and A6) of this paper:

Phase I: A1 first developed a draft coding schema of the categories of search intentions using 100 sessions selected from the sampled 1,086 sessions.[3] To identify the goals of the search sessions, A1 replicated all requests to Stack Overflow by reading all queries and visited all pages in each search session. Then A5 and A6 used the draft coding schema to categorize the goals of the search tasks of 100 sessions collaboratively. During this phase, the draft coding schema of the goals of the search tasks of the 100 sessions was revised and refined. At the end of Stage 1, we obtained 9 categories of search intentions.

Phase II: A1 and A5 applied the resulting coding schema of Phase I to categorize the remaining 986 sessions independently. They were instructed to take notes regarding the deficiency and ambiguity of the coding schema for categorizing certain sessions. The inter-rater agreement (Cohen's kappa) for identifying search intention of this stage is 0.69, indicating that the agreement level is substantial [55].

Phase III: A1, A5, and A6 discussed the coding results obtained in Phase 2 to resolve the disagreements. Disagreements between the three authors are resolved by voting. For example, for the session with one query *difference between session time and current time*, A1 considered that the goal of the session is related to identifying the difference between the terminology *session time* and *current time* in an Oracle database [10]. A5 considered the goal of the session as to calculate the time interval between *session time* and *current time*. We finally considered that the goal of the session is related to calculating the time interval because the developer visited a post related to using time interval to keep track of online users [7]. At the end of Stage 3, we obtained the final coding schema and the final coding results of the sampled 1,086 sessions. Table 7 shows the categories of the search intentions.

After identifying the semantic tags in queries and the categories of the search intentions, we iteratively move along the semantic hierarchy to merge lower semantic tags into a higher semantic tag. However, some of the merges can change the categories of the search intentions. For example, queries with the semantic tag *development tool* are related to the *use of development tool*. Queries with the semantic tag *technology name* are related to several search intentions, e.g., technology combination, function call, excluding

[3] Note that the 100 sampled sessions in phase I in the queries categorization process and the 100 sampled sessions in phase I in the search intentions categorization process are randomly sampled. They are not designed to be the same or different.

Table 6: Categories of semantic tags in queries at the concept level. We also present the description, example, and the proportion of the search sessions that have corresponding semantic tags among the sampled 1086 search sessions.

Category	Description	Example	%
Programming language and technology	The name of the coding technology, e.g., programming language, package name, library name, OS, schema.	**python, angular, pyqtgraph**	61.98%
Task	The action verb refer to the task to be implemented	jQuery **upload** file	42.71%
Data type	The data types that can be assigned with values, entities, etc.	different **colors** in the same plot matplotlib	27.34%
Object	The non-programming terminologies nouns that are predicated by task verbs.	count **vowels** assembly languages	24.74%
Condition	The adjective refer to conditions to the tasks and the objects.	c# **maximim** integer value	23.44%
Method	The callable method of a programming technology in language, library, package, etc.	**heapq.nlargest,**	10.67%
Error information	The bug trace, error code, or description that indicate the code cannot run as expectation.	**[c#] vertical input error**	9.64%
Development tool	The tool that are used to help with coding process, e.g., testing suite, CI/CD tools.	**git** commit -a	7.81%
Question word	The question words in English.	**why, how, what, where**	6.25%
Code	A code snippet.	**atoi(argv[1])**	1.30%

use of development tool. We cannot merge the semantic tag *technology name* with *development tool*. We abandon the merge if the merge can change the categories of the search intentions. By doing so, we generalize semantic tags into 9 more abstract semantic tags.[4]

Results:

4.3.1 Semantic Tags. In terms of the distribution of the number of semantic tags per query, we observe that 32.4% of the sampled queries only have one semantic tag. 31.0% of the sampled queries have two semantic tags. 22.2% of the sampled queries have three semantic tags. Table 6 shows the semantic tags that are identified in Stack Overflow queries at the **concept level**. We observe that **semantic tags related to programming terminologies are the most common in Stack Overflow queries.** More specifically, 61.98% of the sampled search sessions search with programming language names or technology names that are related to their search intentions in at least one query.

However, we observed that 17.41% of the sampled search sessions only search *data type*, *method*, *error information*, and *code*, without specifying the programming languages and their corresponding technologies, e.g., library and framework. It is difficult to identify related programming languages and their corresponding technologies if the Stack Overflow search engine indexes its content using keyword matching. We suggest that Stack Overflow also should take the relationship between programming terminologies in terms of their technical aspect into consideration in its search engine. For example, if the original query only has the function name of a certain library, Stack Overflow could identify the library name and append the library name to queries.

23.44% of the search sessions have at least one query with additional conditions, i.e., they have adjectives that describe additional conditions. For example, a developer issues the query *sql multiple view* on Stack Overflow. They need to create multiple views in SQL. After searching on general search engines, they may identify that the official documentation of SQL server only has knowledge related to creating a view in SQL [2]. They turn to Stack Overflow to seek solutions for the programming problem with additional conditions. We suggest the documentation teams could provide the official solutions to the tasks with additional conditions by supplementing additional information to the documentation.

4.3.2 Search Intention. Table 7 shows the search intention of the sampled search sessions. During our labeling process described in the Approach part, we could identify that **developers use queries with different semantic tags for different types of search intention.** Stack Overflow could provide guidelines for developers to express different types of intention into queries with different semantic tags.

We observe that **developers search the most for manipulating a data type and function of a certain programming language or technology, e.g., library and framework**. More specifically, 19.33% of the sampled search sessions are related to manipulating a data type in a specific programming language or technology, e.g., instantiating a data type or finding an instance with specific characteristics. 18.30% of the sampled search sessions are related to function calls, e.g., calling a specific method or command. One possible reason is that examples presented in official documentation cannot help with the development task. Developers have to search on Stack Overflow for the examples related to manipulating a specific programming technology and the usage examples of methods. We suggest the official documentation could include a section on frequently asked questions.

Searching for how to complete a certain task is second most common searched on Stack Overflow. More specifically, 18.30% of the sampled sessions are related to concrete development tasks. Developers search for how to develop a detailed task given a programming technology. 10.05% of the sampled sessions are related to searching for a road map of a task even without a specific programming technology. They may not know the programming technologies related to the implementation of the task.

> 23.44% of the search sessions have at least one query that has adjectives that describe additional conditions. Developers use queries with different semantic tags for different types of search intention. Attribute names, data type names, function names, and bug trace of a certain programming technology, are the most common in sampled search sessions.

[4]The detailed full coding schemas of semantic tags can be obtained in https://zenodo.org/record/4730525.

Table 7: Categories of search intention. We present the description, the top 3 most common combinations of semantic tags, and corresponding examples, as well as the proportion of the sampled 1086 search sessions that are associated with the intention.

Category	Description	Typical combinations of semantic tags	Example	%
Technology manipulation	Search for how to manipulate programming technologies, e.g., assign values, find the max value.	data type	mappolygon	19.33%
		technology name, data type	ILIST ROW	
		technology name, data type, task	delete row dataframe	
Function call	Search for the use example of certain callable method, API, etc.	technology name	Libgdx	18.30%
		method	heapq.nlargest	
		technology name, method	groupby r	
Concrete development task	Search for how to implement a certain development task in certain programming technology. Developers usually identify such task when coding in editor.	task	order by	18.30%
		technology name, task, object	jQuery upload file	
		task, technology name	jquery capitalize	
Bug fixing	Search for the knowledge related to the bug information presented in console or log.	bug trace	EnvironmentError: mysql_config not found	12.63%
		technology name, bug trace	Japanese Characters are rendering ?? in gridview of asp.net	
		bug summarization	[c#] vertical input error	
General task	Search for how to implement certain task. No programming technology is limited.	object	red-black tree	10.82%
		task, object	compare data	
		condition, object	dynamic watermark	
Use of development tool	Search for the use of programming tools during development process, e.g., coding, testing, version control, delivering.	tool	save data using xml visual studios	8.76%
		tool, data type	atom encoding	
		tool, task	[pydev]Install	
Technology combination	Search for how to use one programming technology with other programming technologies.	technology name, technology name	gcc compiler on apple	5.67%
		technology name, technology name, method	c# netstat	
		technology name, technology name, task	mysql .db file c# connect	
Concept Explanation	Search for the general terminologies.	general terminology	brucelee chamion practise youtube	5.42%
		what technology name	what is parse in android	
Code snippets	Search for the use of code snippets.	code snippets	var port = normalizePort(process.env.PORT || '3000')	0.77%

5 DISCUSSION

5.1 Implications

5.1.1 Actionable Suggestions for Researchers. In Section 4.2, we observe that 71.78% of the reformulations can be fitted into the reformulation strategies identified in general search. 67.88% of the undetermined reformulations can be observed in Huang and Efthimiadis's work. This indicates that developers commonly use the same reformulation strategies in Stack Overflow as in general search. Prior studies have developed tools for query reformulation on Stack Overflow [23, 59]. However, they did not compare the query reformulation process between general search and searching for programming-related knowledge on Stack Overflow. Our research finds the unique features of query reformulation on Stack Overflow. We suggest that future work related to query reformulation on Stack Overflow can propose specific optimization approaches for the types of reformulations that do not appear in general search.

5.1.2 Actionable Suggestions for Stack Overflow. In Section 4.1, we observe that for the sessions that do not end up in posts, only 0.01% of them propose questions on Stack Overflow. This indicates that developers commonly do not ask questions when they cannot find the knowledge related to their programming problems on Stack Overflow. This would result in a lack of related knowledge on Stack Overflow. We suggest that Stack Overflow could survey why users do not ask questions after they cannot find the knowledge related to their search intentions. By determining the root cause, Stack Overflow could propose corresponding solutions.

In Section 4.1, we identify that when developers search from a post, the time interval between requests is the largest. One possible reason is that it takes the most time for developers to identify the similarities and differences between their search intention and the visited posts. They need time to write the next query. We suggest Stack Overflow could enable developers to label which content in the posts is related to their search intention, and which content in the posts is irrelevant to developers' search intention. By doing so, Stack Overflow could propose a tool to help developers to automatically reformulate queries from the visited posts.

In Section 4.3, we observe that 17.41% of the search sessions only search fragments of source code artifacts (e.g., class and method names) without specifying the names of programming languages and technologies. However, programming languages and technologies are covered by the Stack Overflow tag system, while the fragments of source code artifacts are not. It is difficult to identify related programming languages and the corresponding technologies if the Stack Overflow search engine indexes its content using keyword matching. Moreover, in Section 4.2, we observe that for the reformulations that cannot be mapped to the strategies identified in general search, 26.69% of them are related to the domain-specific characteristics, e.g., replacing a programming terminology with another programming terminology that are related in terms of technical

aspects. This indicates that Stack Overflow could not identify the relations between programming terminologies. Developers have to include possible related programming terminologies in their queries. We suggest researchers could set up a database similar to WordNet. The database should include the technical relations (e.g., *is-a* relation [42], *has-a* relation) between all programming terminologies, e.g., class name, data structure, and algorithm name, that are shared on Stack Overflow. This database can represent a more fine-grained programming ecosystem from high-level language and library names down to names of related source code artifacts. By doing so, Stack Overflow could improve the performance of its search engine by considering the terminologies that are related in terms of their technical aspects. Developers do not need to reformulate their queries by trying programming terminologies at different levels of granularity, e.g., replacing a method name with a class name or library name.

In Section 4.1, we identify that when developers search from the home page, the time interval between requests is the second-largest compared with other search activities. In Section 4.3, we observe that developers use queries with different semantic tags for different types of search intention. Although Stack Overflow provides guidelines for developers to ask and answer questions, it does not provide guidelines for developers on how to search [4, 6]. We suggest Stack Overflow could propose guidelines for developers when they search on Stack Overflow, e.g., how to express their search intention into queries.

5.2 Threats to Validity

Threats to internal validity concern the factors that could have influenced our results. We heavily depend on manual processes as described in Section 4.2 and Section 4.3. Like any human activity, our manual labeling process is subject to personal bias. To reduce personal bias in the manual labeling process, each sample was labeled by two of the authors, and discrepancies were discussed until a consensus was reached. We also showed that the level of inter-rater agreement of the qualitative studies is high (i.e., the values of Cohen's kappa ranged between 0.69 to 0.82).

We distinguish different sessions if the time interval between consecutive requests was more than six minutes. However, developers could search for different search tasks in six minutes. In Section 4.2, we observe that 22.54% of the undetermined reformulations are related to the fact that the second query searches for new topics compared with the previous queries. This indicates that 6.21% of search sessions might contain more than one task. Determining whether queries relate to the same search sessions with a time interval of over 6 minutes brings additional threats to our work.

Threats to external validity concern the generalization of our findings. Our study is conducted to investigate the search activities on Stack Overflow. That said, our findings may not be generalized to the search activities in the search engines in other programming-related Q&A websites and general search engines. Stack Overflow covers a wide range of programming-related technologies. Other programming-related Q&A websites, e.g., Google Product Forums[5] and Microsoft Community[6], only share the knowledge that is related to a specific technology. The queries in other programming-related Q&A websites may not need to explain the related technologies. Moreover, the search engines of programming-related Q&A websites (including Stack Overflow) only index the knowledge hosted within the websites. In contrast, general search engines could index the knowledge across the Internet. This indicates that when using the search engines provided by programming related Q&A websites, developers could only search for the knowledge related to the development tasks. However, users could search in general search engines for a wide range of knowledge, e.g., from the use of a certain technique to planning for a trip. Some of the search activities are exclusive to the general search engines, e.g., from the search result list page to the image search engine. We believe our observations can provide insights for the search activities that are common in other search engines and Stack Overflow when searching for programming-related knowledge. Besides, we follow the standard methodologies in the field. We believe this methodology can be generalized to other software engineering platforms (e.g., GitHub) if the search engine logs contain related data (e.g., queries, visited webpages, and timestamps).

6 CONCLUSION

In this paper, we perform a large-scale analysis to characterize how developers search on Stack Overflow. Based on the HTTP requests to Stack Overflow's web servers from December 1, 2017, to November 31, 2018, we observe that reformulating the immediately preceding query is the most frequent search activity on Stack Overflow. To better understand the gap between the knowledge of users and the knowledge available through search engines, we characterize query reformulations and query content. We observe that 71.78% of the reformulations can be fitted into reformulation strategies previously identified by Huang and Efthimiadis for general search activities [33]. The most common reformulation strategies are specializing the search intention, using advanced search mechanisms, and generalizing the search intention. 23.44% of the search sessions have at least one query with adjectives that describe additional conditions. Developers use queries with different semantic tags for different types of search intention. In the future, we plan to propose specific optimization approaches for the reformulation types that do not appear in general search, e.g., replacing a programming terminology with another programming terminology that are related in terms of technical aspects.

ACKNOWLEDGEMENT

The authors would like to thank Stack Exchange Inc. for sharing the dataset. This research was partially supported by the National Science Foundation of China (No. U20A20173), Key Research and Development Program of Zhejiang Province (No.2021C01014), and the National Research Foundation, Singapore under its Industry Alignment Fund – Prepositioning (IAF-PP) Funding Initiative. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Research Foundation, Singapore.

[5] https://productforums.google.com/forum/

[6] https://answers.microsoft.com/en-us/

REFERENCES

[1] [n.d.]. *Catalog Services for the Web (CSW).* https://docs.geoserver.org/stable/en/user/services/csw/index.html

[2] [n.d.]. *CREATE VIEW (Transact-SQL).* https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql?view=sql-server-ver15

[3] [n.d.]. *CSW.* https://en.wikipedia.org/wiki/CSW

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

[5] [n.d.]. *How do I search?* https://stackoverflow.com/help/searching

[6] [n.d.]. *How do I write a good answer?* https://stackoverflow.com/help/how-to-answer

[7] [n.d.]. *PHP event on session end?* https://stackoverflow.com/questions/7226145/php-event-on-session-end

[8] [n.d.]. *reformulationClassifier.* http://jeffhuang.com/reformulationClassifier.py

[9] [n.d.]. *Search.* https://stackoverflow.com/search

[10] [n.d.]. *SESSIONTIMEZONE.* https://docs.oracle.com/database/121/SQLRF/functions176.htm#SQLRF06105

[11] [n.d.]. *Stack Exchange Data Explorer.* https://data.stackexchange.com/

[12] [n.d.]. *Total number of questions and answers for the last 12 months (in 30 day chunks).* https://data.stackexchange.com/stackoverflow/query/6134/total-questions-and-answers-per-month-for-the-last-12

[13] [n.d.]. *What are tags, and how should I use them?* https://stackoverflow.com/help/tagging

[14] [n.d.]. *Why are some questions marked as duplicate?* https://stackoverflow.com/helpcenter/duplicates

[15] Hervé Abdi. 2007. Bonferroni and Šidák corrections for multiple comparisons. *Encyclopedia of measurement and statistics* 3 (2007), 103–107.

[16] Rakesh Agarwal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, Vol. 487. 499.

[17] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. 207–216. https://doi.org/10.1145/170036.170072

[18] Martin Arlitt. 2000. Characterizing web user sessions. *ACM SIGMETRICS Performance Evaluation Review* 28, 2 (2000), 50–63. https://doi.org/10.1145/362883.362920

[19] Sushil Krishna Bajracharya and Cristina Videira Lopes. 2012. Analyzing and mining a code search engine usage log. *Empirical Software Engineering* 17, 4 (2012), 424–466. https://doi.org/10.1007/s10664-010-9144-6

[20] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A Context-aware Time Model for Web Search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16*. ACM Press, Pisa, Italy, 205–214. https://doi.org/10.1145/2911451.2911504

[21] Andrei Z. Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. 2007. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*. ACM Press, Amsterdam, The Netherlands, 231. https://doi.org/10.1145/1277741.1277783

[22] Peter Bruza and Simon Dennis. 1997. Query Reformulation on the Internet: Empirical Data and the Hyperindex Search Engine.. In *RIAO*, Vol. 97. Citeseer, 488–499.

[23] 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. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 1273–1285. https://doi.org/10.1109/ICSE43902.2021.00116

[24] Yi Chang and Hongbo Deng (Eds.). 2020. *Query Understanding for Search Engines*. The Information Retrieval Series, Vol. 46. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-58334-7

[25] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*. 1–10. https://doi.org/10.1145/1526709.1526711

[26] Norman Cliff. 2014. *Ordinal methods for behavioral data analysis*. Psychology Press. https://doi.org/10.4324/9781315806730

[27] Olive Jean Dunn. 1961. Multiple comparisons among means. *Journal of the American statistical association* 56, 293 (1961), 52–64. https://doi.org/10.1080/01621459.1961.10482090

[28] Sonal Gupta, Diana L MacLean, Jeffrey Heer, and Christopher D Manning. 2014. Induced lexico-syntactic patterns improve information extraction from online medical forums. *Journal of the American Medical Informatics Association* 21, 5 (2014), 902–909. https://doi.org/10.1136/amiajnl-2014-002669

[29] Matthias Hagen, Martin Potthast, Anna Beyer, and Benno Stein. 2012. Towards optimum query segmentation: in doubt without. In *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*. ACM Press, Maui, Hawaii, USA, 1015. https://doi.org/10.1145/2396761.2398398

[30] Daqing He, Ayşe Göker, and David J Harper. 2002. Combining evidence for automatic web session identification. *Information Processing & Management* 38, 5 (2002), 727–742. https://doi.org/10.1016/S0306-4573(01)00060-7

[31] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Coling 1992 volume 2: The 15th international conference on computational linguistics*. https://doi.org/10.3115/992133.992154

[32] Sharon Hirsch, Ido Guy, Alexander Nus, Arnon Dagan, and Oren Kurland. 2020. Query reformulation in E-commerce search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1319–1328. https://doi.org/10.1145/3397271.3401065

[33] Jeff Huang and Efthimis N. Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*. ACM Press, Hong Kong, China, 77. https://doi.org/10.1145/1645953.1645966

[34] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. 2009. Patterns of query reformulation during Web searching. *Journal of the American Society for Information Science and Technology* 60, 7 (2009), 1358–1371. https://doi.org/10.1002/asi.21071 _eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.21071.

[35] Bernard J Jansen, Amanda Spink, Chris Blakely, and Sherry Koshman. 2007. Defining a session on Web search engines. *Journal of the American Society for Information Science and Technology* 58, 6 (2007), 862–871. https://doi.org/10.1002/asi.20564

[36] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 193–202. https://doi.org/10.1145/2556195.2556220

[37] William H Kruskal and W Allen Wallis. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association* 47, 260 (1952), 583–621. https://doi.org/10.1080/01621459.1952.10483441

[38] Tessa Lau and Eric Horvitz. 1999. Patterns of search: analyzing and modeling web query refinement. In *UM99 user modeling*. Springer, 119–128. https://doi.org/10.1007/978-3-7091-2490-1_12

[39] Ying Li, Zijian Zheng, and Honghua (Kathy) Dai. 2005. KDD CUP-2005 report: facing a great challenge. *SIGKDD Explor. Newsl.* 7, 2 (Dec. 2005), 91–99. https://doi.org/10.1145/1117454.1117466

[40] Chao Liu, Ryen W White, and Susan Dumais. 2010. Understanding web browsing behaviors through Weibull analysis of dwell time. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 379–386. https://doi.org/10.1145/1835449.1835513

[41] Mehdi Manshadi and Xiao Li. 2009. Semantic Tagging of Web Search Queries. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, 861–869. https://www.aclweb.org/anthology/P09-1097

[42] M. Nassif, C. Treude, and M. P. Robillard. 2020. Automatically Categorizing Software Technologies. *IEEE Transactions on Software Engineering* 46, 1 (2020), 20–32. https://doi.org/10.1109/TSE.2018.2836450

[43] Liming Nie, He Jiang, Zhilei Ren, Zeyi Sun, and Xiaochen Li. 2016. Query expansion based on crowd knowledge for code search. *IEEE Transactions on Services Computing* 9, 5 (2016), 771–783. https://doi.org/10.1109/TSC.2016.2560165

[44] Cole S. Peterson, Jonathan A. Saddler, Natalie M. Halavick, and Bonita Sharif. 2019. A Gaze-Based Exploratory Study on the Information Seeking Behavior of Developers on Stack Overflow. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–6. https://doi.org/10.1145/3290607.3312801

[45] Ana-Maria Popescu and Orena Etzioni. 2007. Extracting product features and opinions from reviews. In *Natural language processing and text mining*. Springer, 9–28. https://doi.org/10.1007/978-1-84628-754-1_2

[46] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How does clickthrough data reflect retrieval quality?. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 43–52. https://doi.org/10.1145/1458082.1458092

[47] Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, Jeff Skowronek, and Linda Devine. 2006. Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen'sd indices the most appropriate choices. In *annual meeting of the Southern Association for Institutional Research*. Citeseer, 1–51.

[48] Caitlin Sadowski, Kathryn T Stolee, and Sebastian Elbaum. 2015. How developers search for code: a case study. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*. 191–201. https://doi.org/10.1145/2786805.2786855

[49] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2006. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '06*. ACM Press, Seattle, Washington, USA, 131. https://doi.org/10.1145/1148170.1148196

[50] Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. 2010. An examination of software engineering work practices. In *CASCON First Decade High Impact Papers on - CASCON '10*. ACM Press, Toronto, Ontario, Canada, 174–188. https://doi.org/10.1145/1925805.1925815

[51] Jun Song, Jun Xiao, Fei Wu, Haishan Wu, Tong Zhang, Zhongfei Mark Zhang, and Wenwu Zhu. 2017. Hierarchical contextual attention recurrent neural network for map query suggestion. *IEEE Transactions on Knowledge and Data Engineering*

29, 9 (2017), 1888–1901. https://doi.org/10.1109/TKDE.2017.2700392

[52] Lynda Tamine, Jesús Lovón Melgarejo, and Karen Pinel-Sauvagnat. 2020. What Can Task Teach Us About Query Reformulations?. In *European Conference on Information Retrieval*. Springer, 636–650. https://doi.org/10.1007/978-3-030-45439-5_42

[53] Raphael Tang, Ferhan Ture, and Jimmy Lin. 2019. Yelling at Your TV: An Analysis of Speech Recognition Errors and Subsequent User Behavior on Entertainment Systems. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 853–856. https://doi.org/10.1145/3331184.3331271

[54] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael AS Potts. 2007. Information re-retrieval: repeat queries in Yahoo's logs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 151–158. https://doi.org/10.1145/1277741.1277770

[55] Anthony J Viera and Joanne M Garrett. 2005. Understanding Interobserver Agreement: The Kappa Statistic. *Family Medicine* 37, 5 (2005), 360–363.

[56] Martin Whittle, Barry Eaglestone, Nigel Ford, Valerie J Gillet, and Andrew Madden. 2007. Data mining of search engine logs. *Journal of the American Society for Information Science and Technology* 58, 14 (2007), 2382–2400. https://doi.org/10.1002/asi.20733

[57] 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

[58] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: Automated generation of answer summary to developers' technical questions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 706–716. https://doi.org/doi/10.5555/3155562.3155650

[59] Neng Zhang, Qiao Huang, Xin Xia, Ying Zou, David Lo, and Zhenchang Xing. 2020. Chatbot4QR: Interactive Query Refinement for Technical Question Retrieval. *IEEE Transactions on Software Engineering* (2020). https://doi.org/10.1109/TSE.2020.3016006

[60] Heinz Züllighoven. 2004. *Object-oriented construction handbook: Developing application-oriented software with the tools & materials approach*. Elsevier. https://doi.org/10.1016/B978-155860687-6/50005-0