Googling for Software Development: What Developers Search For and What They Find

Andre Hora
Department of Computer Science
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
andrehora@dcc.ufmg.br

***Abstract*—Developers often search for software resources on the web. In practice, instead of going directly to websites (e.g., Stack Overflow), they rely on search engines (e.g., Google). Despite this being a common activity, we are not yet aware of what developers search from the perspective of popular software development websites and what search results are returned. With this knowledge, we can understand real-world queries, developers' needs, and the query impact on the search results. In this paper, we provide an empirical study to understand what developers search on the web and what they find. We assess 1.3M queries to popular programming websites and we perform thousands of queries on Google to explore search results. We find that (i) developers' queries typically start with keywords (e.g., Python, Android, etc.), are short (3 words), tend to omit functional words, and are similar among each other; (ii) minor changes to queries do not largely affect the Google search results, however, some cosmetic changes may have a non-negligible impact; and (iii) search results are dominated by Stack Overflow, but YouTube is also a relevant source nowadays. We conclude by presenting detailed implications for researchers and developers.**

***Index Terms*—Software Development, Web Search, Google Search, Empirical Study**

I. Introduction

Developers often search for software resources (*e.g.,* code examples, documentation, bug-fixes, etc.) on the web to support development activities [1]–[5]. Prior work reports that developers may spend up to 20% of their time on this task [6]–[8]. For this purpose, many websites are available [9]. Stack Overflow [10], for instance, receives over 50 million users per month and is the 44th most visited website in the world. W3Schools [11], another popular website to support software development, had 2.5 billion pageviews in 2019 and is among the top 200 most visited ones. Typically, developers rely on web search engines, such as Google, to find software resources on the web [1]–[5], [8], [12]–[14]. That is, instead of going directly to those websites, developers go firstly to search engines to perform their search queries. According to Alexa [15], Stack Overflow and W3Schools have over 85% of their traffic coming from web search engines.

When searching on the web, developers are free to type in natural language, producing a rich log with the queries they are performing. In the literature, some studies assess those search queries to learn from them. Bajracharya *et al.* [16], [17] mined the log usage of a code search engine, Koders, to understand what their users are searching for. Sadowski *et al.* [18] performed a case study at Google to learn how developers search for code and the properties of the queries. Those research studies have a special focus on code search. However, code search is not the sole goal of developers when navigating on the web: they may also search for a variety of other resources [5], such as novel technologies, debugging logs, and installation guides, to name a few. In this context, Rahman *et al.* [14] assessed the queries of 310 developers and built a model to detect code and non-code related queries. Xia *et al.* [5] analyzed the queries from 60 developers to better understand their search tasks. Recently, Microsoft researchers [19] proposed a classifier for distinguishing software engineering dedicated queries from other queries.

To the best of our knowledge, the literature has not yet explored what developers search on the web *from the perspective of the most popular websites dedicated to software development*. That is, what queries performed on web search engines such as Google lead to websites such as Stack Overflow and W3Schools, and what search results are returned to developers. This knowledge can be used to understand real-world search queries and developers' needs, thus, supporting the creation of better tools and targeted websites. Assessing the search results can help to detect where search engines find software resources and how the search queries affect the search results, fomenting the discovery of novel sources and the creation of better searching tools for developers.

In this paper, we provide an empirical study to better understand what developers and programmers search on the web and what they find. First, we assess 1.3M **search queries** to five popular programming websites: Stack Overflow, W3Schools, GeeksForGeeks, TutorialsPoint, and ProgramCreek; this data is collected from alexa.com [15]. We then propose the following research questions to study developers' search queries:

- *RQ1 (query content): What is the content of the search queries?* Developers' queries reference key contexts, such as programming languages (30%), software technologies (24.5%), and web frameworks (5%). They mostly perform code reuse (45.8%) and general (20.1%) searches.
- *RQ2 (query size and keyword position): What is the size of the search queries? Where are the keywords located?* Developers' queries have 3 words on the median, which is similar to general web search queries. Queries typically start by referencing a technology (*e.g.,* "*c# read file*").

TABLE I
SELECTED SOFTWARE DEVELOPMENT WEBSITES.

Websites	Short Description	Traffic Rank	Indexed Pages	Web Traffic	Queries	Top 3 Words
stackoverflow.com	Question and answer platform for developers	44	35M	94%	500,014	python, php, javascript
w3schools.com	Website with code tutorials, references, and examples	177	47K	85%	478,537	javascript, html, css
geeksforgeeks.org	Computer science portal with articles and tutorials	331	200K	92%	476,403	python, java, javascript
tutorialspoint.com	Education portal with courses, tutorials, and examples	485	170K	90%	500,000	python, java, c
programcreek.com	Programming website with code examples	9,153	690K	95%	90,085	python, java, example

- *RQ3 (query structure): How are the search queries structured?* Developers tend to omit function words (*e.g.,* conjunctions) in queries, which are mostly composed of content words (*e.g.,* nouns and verbs). The ratio is similar to general web search queries: 20% x 80%.
- *RQ4 (query similarity): How similar are the search queries?* Most of the developers' queries are similar among each other: 60% have at least one similar peer, whereas 8% have 10 or more similar ones.

Next, based on the insights from the developers' queries, we perform thousands of queries on Google (with the Google Search API [20]) to assess and explore **search results**:

- *RQ5 (result resources): Where does Google find software resources?* Google finds software resources mostly on stackoverflow.com (11%), youtube.com (6%), and w3schools.com (5%). However, the results tend to vary according to query type (*e.g.,* more or less general).
- *RQ6 (result variation): How do Google results vary for similar queries?* Overall, performing minor changes to queries (*e.g.,* swapping, omitting, and using synonymous words) do not broadly affect the top 1 search results nor the overall top 10. However, there are a few exceptions, *e.g.,* some word swap may have a non-negligible impact.

Based on our findings, we provide implications for developers and researchers: (1) we reveal novel empirical data that can guide software vendors and content providers understanding developers' queries and needs; (2) we shed light on the existence of query patterns specific to development, which can be useful for research areas as automated query suggestion and reformulation [21]–[23]; (3) we present that the overall stability of the Google search results can foment programming tools that rely on web search, *e.g.,* tools to find errors/exceptions [24]–[27]; (4) we reveal that minor cosmetic query changes can affect the Google results, indicating room for improvement in the software search landscape; and (5) we bring to light the priority of Google search to Stack Overflow.

This paper has the following contributions:

- We provide a large study to characterize real-world *search queries* performed by developers on the web (Section III).
- We provide a large analysis to understand the Google *search results* due to developers' queries (Section IV).
- We make our Google search results publicly available to the research community (Section IV).
- We propose implications and insights for developers and researchers to improve programming tools and foment novel researches (Section V).

II. STUDY DESIGN

A. Selecting the Websites

We select five very popular websites to perform this study: Stack Overflow [10], W3Schools [11], GeeksForGeeks [28], TutorialsPoint [29], and ProgramCreek [30]. They are well-known platforms that support software development, as presented in Table I. Stack Overflow, for example, is a question and answer platform, whereas TutorialsPoint is an education portal that includes tutorials. Column "Traffic Rank" presents their ranking position worldwide as measured by alexa.com [15]. Those websites also have thousands of webpages indexed by the Google search engine, receiving millions of users per month (*e.g.,* Stack Overflow has 51M users/month, while TutorialsPoint 40M). Column "Web Traffic" shows the ratio of users coming from web search traffic (measured by Alexa), that is, web search engines, such as Google, Yahoo!, Bing, etc. The majority of the users come from web searches, ranging from 85% in W3Schools to 95% in ProgramCreek.

B. Collecting the Search Queries

We analyze the search queries performed by developers on web search engines that lead to the selected websites. We rely on Alexa to collect the search queries. One can log in directly into alexa.com and download the queries per website; we collected it in June 2020. Alexa traffic estimates are based on data from global traffic, which is a sample of millions of internet users [15].

Table I shows the number queries in each dataset (column "Queries"), which varies from 90,085 in ProgramCreek to 500,014 in Stack Overflow. After grouping the queries and removing duplicated ones, we find **1,306,628** distinct queries. The last column presents the most frequent words in the queries: as expected, they are in the context of software development, such as Python, JavaScript, and HTML. Our dataset is publicly available at https://doi.org/10.5281/zenodo.4429258.

C. Research Questions: What Developers Search For

1) RQ1: Query Content: In our first research question, we assess the context and intent of the developers' queries.

Query context. First, we assess the context of the search queries to acquire insights on what developers are looking for on the web. As presented in Table II, we explore the query context regarding three categories: programming languages, web frameworks, and popular technologies.

Specifically, we analyze the queries and look for references to the 100 most popular programming languages (according

TABLE II
QUERY CONTEXT CATEGORIES.

Context	#	Examples
Programming lang.	100	C, Java, Python, C++, C#, Visual Basic, JavaScript, PHP, SQL, R
Web frameworks	100	React, ASP NET, Angular, Rails, Vue, Django, Laravel, Spring, Express, Flask
Other technologies	53	JQuery, Ajax, Android, Selenium, Numpy, Pandas, Bootstrap, D3, CSS, HTML

to the TIOBE Index)[1] and to the 100 most popular web frameworks (according to the Hot web framework ranking).[2] Examples of queries with references to languages and web frameworks include "*read from json file in c#*", "*example of python tuple*", and "*when to use react*". Besides, we also look for references to other 53 popular software technologies. This set was manually curated by inspecting the most frequent words in our dataset of queries. It includes a miscellany of software libraries, frameworks, and tools, such as JQuery, Android, and Selenium (*e.g.,* "*splash screen in android*"). *Rationale*: we aim to assess whether developers are likely to mention the underlying software technologies behind the queries. Queries including those references are more specific and possibly easier to be resolved by the search engine. For example, "*arraylist add*" is dubious and may refer to both Java and C#, while "*java arraylist add*" may produce better results.

Query intent. We rely on the categories proposed by Xia *et al.* [5], who performed a similar analysis, to assess developers' search tasks. This has two advantages: (1) we keep consistency with previous research studies and (2) we can directly compare our results with previous ones. Table III presents the seven query categories: general search includes general purpose queries (*e.g.,* "*lifecycle of android*"); debugging and bug fixing are about exceptions, errors, and bugs (*e.g.,* "*curl: failed to connect*"); programming category contains queries mostly related to programming language features and design patterns (*e.g.,* "*protected field in java*"); third party code reuse includes queries about APIs and reusable examples (*e.g.,* "*javascript date get year*"); tools contain tool-related queries (*e.g.,* "*git delete files*"); database is about search queries related to SQL statements (*e.g.,* "*sql drop all foreign keys*"); and testing contains queries in the context of software testing (*e.g.,* "*what is a smoke test*"). Since the categories are well-defined, the query classification is straightforward. In this analysis, we randomly select 384 queries (*i.e.,* 95% confidence level and 5% confidence interval) and manually classify them. *Rationale*: developers may search on the web for a variety of reasons, from code [16]–[18] to other resources (*e.g.,* novel technologies, etc.) [5]. However, it is not clear what are their major concerns from the perspective of the studied websites, which are among the most popular nowadays.

[1]https://www.tiobe.com/tiobe-index
[2]https://hotframeworks.com

TABLE III
QUERY INTENT CATEGORIES (BASED ON [5]).

Intent	Short description
General search	Technologies, terminologies, background knowledge, best practices, laws or regulations, licenses, or any generic software terms
Debugging and bug fixing	Exceptions/error messages, bugs (*e.g.,* programming, configuration, security, performance, multi-threading, etc.)
Programming	Programming language features and design patterns
Third party code reuse	APIs, reusable examples, libraries, frameworks, HTML/CSS
Tools	command line, SO, IDEs, version control systems, issue tracking systems
Database	SQL statements, no-SQL database, optimization solutions
Testing	Testing methods, automated testing tools, public testing datasets

2) RQ2: Query Size and Keyword Position: We now assess the size of the queries and the position of the contextual words.

Query size. As typically adopted in the literature [4], [18], [31], we compute query size in terms of the number of words. *Rationale*: general web search queries are known to be short [31], while code search queries may vary [4], [17], [18]. While shorter queries may point to broader issues, longer ones may reveal the need for the resolution of specific problems.

Keyword position. We also assess the position of contextual words (the 253 of Table II) in queries with size ≥ 3 (that is, three or more words). *Rationale*: we aim to better understand where contextual words are located in the search queries, that is, whether developers are likely to position such keywords, in the beginning, middle, or end of the query. This assessment provides the basis for us to analyze whether word position may affect the search results, which we further explore in RQ6.

3) RQ3: Query Structure: In this research question, we investigate the structure of the developers' queries.

Part of speech. Words have distinct roles in natural language, with a distinction between *content words* (nouns, verbs, adjectives, and adverbs) and *function words* (prepositions, conjunctions, etc.) [32]. Content words represent the core information in a sentence and can be meaningful in isolation because of their straightforward semantics (*e.g.,* "JavaScript") [33]. In contrast, function words represent the relationships between content words and often do not carry relevant meaning in isolation (*e.g.,* "the") [33]. Despite the set of function words is smaller than the set of content words, function words are more frequently used than content words, representing over half of the words we use in daily speech [33]–[35]. They both compose the part of speech (POS) in a sentence. We apply part of speech tagging to assess the query structures. *Rationale*: users have a mental model of what search engines can handle, thus, most users tend to omit function words in search queries, possibly believing they are less important in query effectiveness [32], [36]. We explore the most common tags (noun, verb, conjunction, etc.) in developers' queries and

whether they follow such a mental model, which is common in general web search [32], [36]. Also, after gaining insights in this analysis, we explore in RQ6 whether function words and synonymous verbs are likely to impact the search results.

Other frequent terms. We analyze the presence of other frequent terms in the queries. First, we inspect the presence of the five Ws (who, what, where, when, and why) and how, also known as 5w1h,[3] which represent questions whose answers are considered basic in information gathering or problem-solving (*e.g.,* "*how to zip in linux*"). We also assess other frequent terms found in the part of speech analysis. *Rationale*: we aim to understand to what extent and how frequently some terms are adopted in software development queries. We further assess how those terms affect the search results in RQ6.

4) RQ4: Query Similarity: We explore how similar are the search queries. For this purpose, we evaluate three similarity metrics: Cosine [37], Jaccard [38], and weighted Jaccard [39].[4] These metrics are typically adopted in information retrieval and code search literature to compare the similarity between two documents [8], [41], [42]. *Rationale*: we aim to discover to what extent developers create similar queries possibly to resolve similar problems and to reveal those small variations.

Typically, the adopted metric and threshold vary according to the target problem. To find the best option for our problem, we create an oracle of similar queries and evaluate each combination of metric and threshold, as follows.

(i) Creating an oracle of similar queries. First, we select 10 base queries in distinct technologies, including Java, Python, C#, JavaScript, PHP, HTML, and CSS (see Table IV). For each base query, we collect from our dataset all the queries in which the base query is included. For instance, considering the first query "*java check string is number*", we collect 12 queries, such as "*check if a string is a number java*". Considering the 10 base queries, we automatically collect a total of 175 similar queries, which is our oracle.

TABLE IV
QUERY SIMILARITY BEST RESULT (JACCARD, THRESHOLD: 0.8)

Base query	#	TP	FP	FN	Prec	Rec	F1
java check string is number	12	12	1	0	0.92	1.0	0.96
java comparable vs comparator	9	9	0	0	1.0	1.0	1.0
java private public protected	13	13	0	0	1.0	1.0	1.0
python get file size	12	10	0	2	1.0	0.83	0.91
python check item in list	12	12	1	0	0.92	1.0	0.96
c# reverse string	14	12	0	2	1.0	0.86	0.92
javascript break loop	48	33	0	15	1.0	0.69	0.82
php array remove duplicates	18	16	0	2	1.0	0.89	0.94
html change color font	19	16	0	3	1.0	0.84	0.91
css zoom animation	18	16	0	2	1.0	0.89	0.94
Total	175	149	2	26	0.99	0.85	0.91

(ii) Detecting the best similarity metric and threshold. We first split our query dataset according to their technologies (the same of Table II), including only queries with three or more words (query size ≥ 3). Thus, the queries are grouped according to their technologies and we avoid very small and meaningless queries. For each split dataset, we run the three similarity metrics using distinct threshold values, ranging from 0.5 to 0.9 by increments of 0.1, totaling 15 combinations. As often performed in information retrieval, we tokenize and remove stop words for this analysis. Therefore, for each combination, the queries are grouped in *similar queries* according to the evaluated similarity metric and threshold. Then, for each combination, we collect the groups of similar queries that include the base queries. Finally, we assess the accuracy of the collected groups of similar queries against the oracle, in terms of precision, recall, and F1 score [43]. As detailed in Table IV, the best result in terms of F1 score (*i.e.,* the harmonic mean of the precision and recall) occurred for the metric Jaccard and threshold 0.8. This combination resulted in an overall precision of 0.99, recall of 0.85, and F1 score of 0.91. Thus, in this study, we adopt the Jaccard metric with a threshold of 0.8 to group similar queries.

D. Research Questions: What Developers Find

In the following RQs, we focus on the search results.

1) RQ5: Google result resources: To understand what developers find on the web, we first assess the returned websites for their queries. For this purpose, we rely on the official Google Search API [20] to programmatically search on Google [44]. We selected Google because it is the *de facto* web search engine nowadays with over 92% of the search market share [45]. We perform two types of queries (specific and generic), which are randomly selected from our query dataset. The specific queries have references to popular technologies: Python, JavaScript, Java, PHP, and CSS. The generic queries include the broader terms *how to*, *what is*, *example*, *vs*, and *convert to*, which come from RQ3. For each term, we select 500 queries, totaling 5K queries. Lastly, from the Google search results, we collect the top 10 returned links and analyze their source websites. *Rationale*: detecting software resources on the web informs software vendors where developers find information about their products, it notifies developers where they can spot software information, and it supports researchers to study the software resource landscape on the web [9], [46].

2) RQ6: Google result variation: Lastly, we explore the variation of the search results due to similar queries. *Rationale*: we aim to understand whether Google search results may change due to minor query changes likely done by developers. This way, we become aware of missing resources and its possible impact on developers and programming tools that rely on web search (*e.g.,* [24]–[27]). Next, we detail this evaluation.

(i) Modifying the search query. We propose 28 modification types, which are grouped into four categories: word swap, word removal, synonymous word, and similar query, as detailed in Table V. In *word swap*, we swap a target word from the beginning to the end of the query; in *word removal*, we remove a target word; in *synonymous word*, we replace a target word by a synonym;[5] and in *similar query*, we select queries

[3] https://en.wikipedia.org/wiki/Five_Ws

[4] We use TF-IDF in the weighted Jaccard to represent the weights [40].

[5] Based on Stack Overflow synonyms for tags: stackoverflow.com/tags.

from the groups of similar queries in RQ4 (*i.e.,* Jaccard, threshold: 0.8). Note that some changes are simple, *e.g.,* from the original query "*python email parser*" to the modified version "*email parser python*". In contrast, other changes may affect the query meaning, *e.g.,* from "*convert txt to json*" to "*convert json to txt*". For each modification type, we randomly select (at most) 500 queries and create their 500 modified versions, thus, generating close to 28K queries in total.

(ii) Searching on Google and assessing the results. Lastly, with the Google Search API [20], we search for each query pair (*i.e.,* the original and modified versions) and we assess the top 10 returned links. For each pair of search results, we compare whether the top 1 and the top 10 links are exactly the same; we also compute the intersection of links in the top 10.

TABLE V
QUERY MODIFICATION SEARCHED ON GOOGLE.

Cat.	Modification Type	Example
Word swap	1. context	python email parser → email parser python
	2. how to	how to zoom in css → zoom in css how to
	3. what is	what is virtualenv → virtualenv what is
	4. example	example xml file → xml file example
	5. vs	map vs list → list vs map
	6. convert to	convert txt to json → convert json to txt
Word removal	7. context	python email parser → email parser
	8. how to	how to zoom in css → zoom in css
	9. what is	what is virtualenv → virtualenv
	10. example	example xml file → xml file
	11. vs	map vs list → map list
	12. convert to	convert txt to json → txt to json
	13. stop words	from bootstrap 3 to 4 → bootstrap 3 4
Synonymous word	14. python → py	python email parser → py email parser
	15. javascript → js	xml encode javascript → xml encode js
	16. java → jdk	weak reference java → weak reference jdk
	17. php → php5	swicth case php → swicth case php5
	18. css → cascading style sheets	twitter css library → twitter cascading style sheets library
	19. get → obtain	vuejs get base url → vuejs obtain base url
	20. create → build	swift create json → swift build json
	21. remove → delete	list remove all → list delete all
	22. check → verify	date format check → date format verify
	23. convert → transform	convert nan to zero → transform nan to zero
Similar query	24. python	python list remove item → how to remove item from list python
	25. javascript	javascript tab open → open tab javascript
	26. java	java char array string → java convert char array to string
	27. php	php get array key → how to get array key php
	28. css	css change color → change color in css

III. WHAT DEVELOPERS SEARCH FOR

RQ1: What is the content of the search queries?

Query context. Table VI summarizes the context of the search queries. Close to 30% (389,185 out of 1,306,628) of the developers' queries have references to programming languages. The top 10 programming languages account for the majority of the cases (26.7%). The most referenced languages are Python (6.2%), JavaScript (5.6%), and Java (4%). In contrast, references to web frameworks are less common: only 5.1% of the queries mention them. We also observe that developers' queries also mention other technologies, such as HTML (3.6%), CSS (3.3%), and JQuery (2.1%). In total, the 53 tracked technologies are referenced by 24.5% of the queries. Altogether, developers are likely to cite the software technologies behind the queries, making their contexts explicit.

TABLE VI
CONTEXT OF THE QUERIES. PY: PYTHON. JS: JAVASCRIPT. T10: TOP 10.

Pos	Prog. languages			Web frameworks			Other technologies		
	Name	#	%	Name	#	%	Name	#	%
1	py	81,027	6.2	react	13,607	1.0	html	46,867	3.6
2	js	73,517	5.6	laravel	13,100	1.0	css	42,805	3.3
3	java	52,033	4.0	angular	12,913	1.0	jquery	27,982	2.1
4	php	45,720	3.5	django	5,450	0.4	bootstrap	23,830	1.8
5	sql	30,438	2.3	spring	5,278	0.4	android	20,010	1.5
T10	-	349,383	26.7	-	61,762	4.7	-	227,296	17.4
All	-	389,185	29.8	-	66,408	5.1	-	320,301	24.5

When analyzing per programming website, we notice only a small difference in the top-1 results (Table VII). Python is the top-1 language in all websites, but W3Schools. The web frameworks vary per website, but they are all among the top-5 of the overall analysis. Lastly, the other technologies are dominated by HTML and JQuery.

TABLE VII
CONTEXT OF THE QUERIES PER WEBSITE (TOP-1).

Websites	Prog. languages	Web frameworks	Other technologies
stackoverflow.com	python	laravel	jquery
w3schools.com	javascript	angular	html
geeksforgeeks.org	python	react	jquery
tutorialspoint.com	python	angular	html
programcreek.com	python	django	android

Query intent. To get more insights into the developers' search tasks, we perform a manual classification of 384 randomly selected queries. Table VIII shows that developers mostly perform code reuse (45.8%) and general (20.1%) searches. Those queries are followed by programming (9.4%), debugging (7.8%), database (7.6%), and tool searches (6.3%). We could not find testing queries in our selection. Xia *et al.* [5] found debugging, code reuse, and general searches as the most common, whereas testing as the least common. The difference in the top searches may happen because, in the prior study [5], the queries belong to specific developers, which work in large outsourcing companies mostly in Java. In our case, the queries are much broader: the developers may have distinct expertise levels and the queries are not specific to any language.

TABLE VIII
INTENT OF THE QUERIES.

Intent	#	%	Examples
Third party code reuse	176	45.8	javascript read line from file
General search	77	20.1	neural network tutorial pdf
Programming	36	9.4	switch case cpp
Debugging and bug fixing	30	7.8	read_csv not working
Database	29	7.6	group by statement sql
Tools	24	6.3	git commit windows
Undefined	12	3.1	two columns
Total	384	100	-

RQ1 Conclusion: Developers' queries typically provide references to key contexts, such as programming languages (30%), software technologies (24.5%), and web frameworks (5%). Developers mostly perform code reuse (45.8%) and general (20.1%) searches.

RQ2: What is the size of the search queries? Where are the keywords located?

Query size. Figure 1 presents the distribution of the size for the 1,306,628 queries. The median number of words per query is only 3 (the first quartile is 3 and the third quartile is 4). Breaking the queries by context, we notice a small variation: queries with references to languages and other technologies are slightly larger (4 words), while web framework queries continue with 3. The literature reports that general web queries are also short: 2.8 words [31]. However, code search studies have found distinct values, *e.g.,* 1.31 [17], 1.85 [18], and 4.2 [4]. Our results concur with those in the sense that some variation in size is expected depending on the query context.

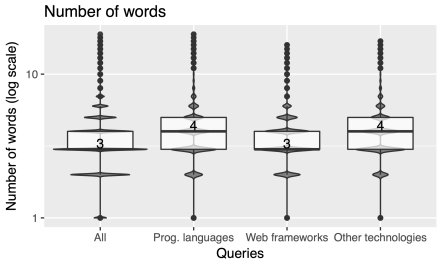


Fig. 1. Distribution of the size of the queries.

Keyword position. We now focus on the queries that include references to languages, web frameworks, and other technologies. We assess where those keywords are located in the query, *i.e.,* first, middle, or last words. As presented in Figure 2, the keywords are mostly the first of the query. In queries with references to languages, 49.2% start by mentioning the language itself (*e.g.,* "*python get file size*"), while 36.9% end with it (*e.g.,* "*get file size python*"); in 13.9%, the reference appears in the middle. In web framework queries, the difference is larger: 65.2% start with the keyword. Lastly, other technology queries also follow a similar trend (first: 48.7%).

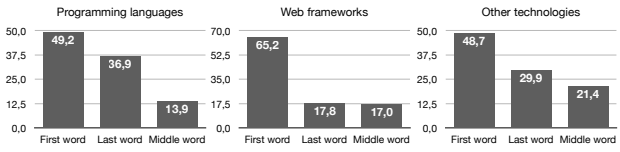


Fig. 2. Position of the keywords words in the queries (query size $\geq$ 3).

RQ2 Conclusion: Developers' queries are short: they have 3 words on the median, which is similar to general web search queries. Keywords are more likely to be the first than the last word in the query.

RQ3: How are the search queries structured?

Part of speech. We present in Table IX the tags used for labeling part of speech in the queries. We notice that the queries are mostly formed by *content words* (80.3%), while *function words* represent only 19.7%. This rate is very similar to the ones found in a recent study about general web search queries [32], which detected 79% for content words and 21% for function words. Thus, as in general web search, this suggests that developers also tend to omit function words in the analyzed search queries, possibly believing they will not affect the query results [32], [36].

TABLE IX
TAGS USED FOR LABELING PART OF SPEECH IN THE QUERIES.

Class	Tag	#	%	Examples
Content	Noun	3,085,194	63.1	python, javascript, php
	Verb	441,435	9.0	create, find, add
	Adjective	271,106	5.5	multiple, new, empty
	Adverb	91,836	1.9	how, where, online
	Total	3,923,167	80.3	-
Function	Adposition	337,616	6.9	in, to, of
	Conjunction	57879	1.2	and, or but
	Other	567965	11.6	to, not, s
	Total	963,460	19.7	-

We also observe in Table IX that the most common content tag is *noun* (63.1%), followed by *verb* (9%) and *adjective* (5.5%). In a related study, Biega *et al.* [32] also found the same order (nouns, verbs, and adjectives) in general web search queries, but with distinct ratios (47%, 15%, and 13%, respectively). This suggests that nouns are more common in developers' queries than in general web search queries, whereas the opposite happens to verbs and adjectives.

We detail in Table X the queries starting with Python, JavaScript, and PHP (the most common starting nouns) and their top 2nd and 3rd words. We notice that 42,608 queries start with *python*, 25,436 with *javascript*, and 23,996 with *php*. Those proper nouns are mostly followed by verbs (*i.e., get*, *print*, and *check*) or common nouns representing data types (*i.e., list* and *array*). Interestingly, the verb *get* is dominant in the three cases, meaning that developers are likely to create queries in the format: <*technology*> *get* <*complement*>, as in "*python get file*" and "*javascript get element*".

TABLE X
QUERIES STARTING WITH PYTHON, JAVASCRIPT, AND PHP.

1st python (42,608)		1st javascript (25,436)		1st php (23,996)	
2nd	3rd	2nd	3rd	2nd	3rd
get (1,010)	current (58) all (41) file (37)	get (1,422)	element (83) current (71) all (47)	get (1,384)	current (76) file (56) first (54)
list (962)	of (85) to (63) remove (33)	array (721)	to (40) of (38) push (32)	array (725)	to (64) remove (31) get (28)

print (769)	list (24)
	to (22)
	format (19)
		check (658)	if (381)
			for (38)
			string (15)
				check (562)	if (303)
					string (23)
					array (22)

Other frequent terms. To better understand how the queries are formed, we assess frequent terms in the queries (Table XI). We also present key accompanying words in those queries, which were manually curated from their most frequent words (after removing proper nouns, such as Python, JavaScript, etc.). We first focus on the 5w1h terms, which represent questions about information gathering or problem-solving. The 5w1h terms happen in 4.7% of the queries: the most common terms are *how* (3.1%) and *what* (1.4%). Notice that their accompanying words are distinct, involving themes as usage, creation, difference, and installation, to name a few.

Next, we analyze the other three terms: *example*, *vs*, and *convert to*, which are among the most commons nouns, adpositions, and verbs, but are very specific to software development. Developers mostly seek examples with terms as file, json, api, post, and class, *e.g.,* "*valid json example*" and "*generic class example*". The term *vs* is used in comparison queries and happens with join, class, list, static, and array, *e.g.,* "*inner join vs join*" and "*vector vs list*". Lastly, the term *convert to* is more attached to data types, such as string, array, int, date, and json, indicating developers are interested in conversions, *e.g.,* "*convert string to decimal*" and "*convert json to csv*".

TABLE XI
FREQUENT TERMS IN THE QUERIES.

Term	#	%	Top Words
5w1h	61,269	4.7	-
- how	40,092	3.1	to, use, create, make, file
- what	18,164	1.4	is/are, mean, between, difference, data
- where	1,269	0.1	is/are, stored, installed, put, file
- why	971	0.1	is/are, use, important, used, not
- when	603	<0.1	to, use, is, vs, class
- who	170	<0.1	is, created, developed, used, invented
example	21,793	1.7	file, json, api, post, class
vs	17,088	1.3	join, class, list, static, array
convert ... to ...	9,034	0.7	string, array, int, date, json

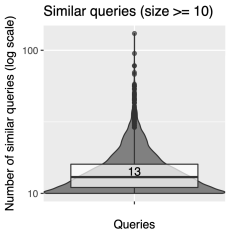
RQ3 Conclusion: As in general web search, developers also tend to exclude function words in their queries, which are mostly composed of content words (80.3%). Developers' queries may make explicit their goals, using terms as 5w1h, example, vs, and convert.

RQ4: How similar are the search queries?

After grouping the queries by applying the Jaccard similarity, we separate the similar queries in three groups (Table XII): the ones with only one query (69%), groups with between 2 and 9 similar queries (30%), and groups with 10 or more queries (1%). In total, 370,001 groups are created with 635,943 queries: 40% of the queries are grouped alone (*i.e.,* group $size = 1$), while 60% have similar queries (*i.e.,* they are in groups with $size \geq 2$). We observe that 50,104 (8%) queries belong to the 3,392 groups with 10 or more similar queries. The right side of the table details this distribution: on the median, those 3,392 groups have 13 similar queries (the first quartile is 10 and the third quartile is 16). This suggests that developers are likely to perform similar queries with small variations. For example, the largest group in our analysis includes 131 tiny queries related to the C programming language, *e.g.,* "*or in c*" and "*for in c*". The second largest group has 95 similar queries related to the usage of array in JavaScript, *e.g.,* "*array in javascript*" and "*if array javascript*". Lastly, the third largest group includes 78 HTML queries related to the opening of new tabs, *e.g.,* "*html open new tab*" and "*new tab open html*".

TABLE XII
STATISTICS OF THE SIMILAR QUERIES.

Groups of Similar Queries			Queries	
Size	#	%	#	%
1	256,899	69	256,899	40
2 to 9	109,710	30	328,940	52
$\geq$ 10	3,392	1	50,104	8
Total	370,001	100	635,943	100



RQ4 Conclusion: Most of the developers' queries are similar among each other: while 40% have no similar counterpart, 60% have at least one similar peer. We also find that 8% have 10 or more similar ones.

IV. WHAT DEVELOPERS FIND

Based on our findings so far, we propose two novel analyses to explore and reason about search results.

RQ5: Where does Google find software resources?

We perform queries on Google to discover the websites it finds software resources on the web. Specifically, we run 5K queries with the support of the Google Search API [20]. Table XIII summarizes this analysis: we find 7,303 distinct websites and 49,203 distinct links. Overall, when considering all the search results (*i.e.,* top 10), the most recurrent websites are stackoverflow.com (11%), youtube.com (6%), and w3schools.com (5%). Interestingly, YouTube is the second one before websites dedicated to software development. Notice that when considering only the top 1 websites, Stack Overflow is even more dominant with 28% of the links. We also observe that 4 out of the 5 studied websites (Stack Overflow, W3Schools, GeeksForGeeks, and TutorialsPoint) appear in the top search results: this is expected because they are among the most popular ones nowadays.

Next, we break the results by keyword and general queries (Table XIV). The keyword queries are about popular contexts found in RQ1 (*e.g.,* Python, JavaScript, etc.), whereas the general ones include common search terms found in RQ3 (*e.g.,* how to, what is, etc.). The keyword queries return fairly less distinct websites than the general ones (column #). For example, the *python* queries returned 764 distinct websites, while the *how to* queries returned almost twice

TABLE XIII
OVERVIEW OF THE MOST RETURNED WEBSITES BY GOOGLE.

Top 10 (all)	#	%	Top 1	#	%
stackoverflow.com	5,246	11	stackoverflow.com	1,393	28
youtube.com	2,931	6	w3schools.com	705	14
w3schools.com	2,656	5	geeksforgeeks.org	217	4
geeksforgeeks.org	2,241	5	php.net	211	4
tutorialspoint.com	1,161	2	en.wikipedia.org	119	2
developer.mozilla.org	1,142	2	developer.mozilla.org	97	2
medium.com	776	2	docs.python.org	96	2
en.wikipedia.org	743	2	docs.oracle.com	89	2
php.net	701	1	tutorialspoint.com	80	2
Distinct websites	7,303			868	
Distinct links	49,203			4,924	

more (1,412). Besides, in addition to dominant websites like stackoverflow.com and w3schools.com, the keyword queries reveal dedicated websites, for example, the *python* queries go to docs.python.org, *javascript* to developer.mozilla.org, *java* to docs.oracle.com, *php* to php.net, and *css* to css-tricks.com.

TABLE XIV
RETURNED WEBSITES BY GOOGLE: KEYWORD AND GENERAL QUERIES.

Query	#	Top 3 websites
python	764	stackoverflow.com, docs.python.org, geeksforgeeks.org
javascript	831	stackoverflow.com, w3schools.com, developer.mozilla.org
java	729	stackoverflow.com, geeksforgeeks.org, docs.oracle.com
php	902	stackoverflow.com, php.net, w3schools.com
css	837	w3schools.com, stackoverflow.com, css-tricks.com
how to	1,412	stackoverflow.com, youtube.com, w3schools.com
what is	1,577	youtube.com, en.wikipedia.org, stackoverflow.com
example	1,175	w3schools.com, en.wikipedia.org, youtube.com
vs	1,751	stackoverflow.com, youtube.com, medium.com
convert to	1,321	stackoverflow.com, youtube.com, geeksforgeeks.org

Another observation is that stackoverflow.com is the top 1 in most cases (7 out of 10), but not for *css*, *what is*, and *example* queries. Lastly, it is worth to notice that youtube.com is top 3 in all general queries, but in none of the specific ones. Examples of general queries in which YouTube is top-ranked include: "*how to use colab*", "*what is real time database*", and "*example of transitive relation*", to name a few. This reinforces YouTube as an important source of knowledge for developers nowadays, and, depending on how the query is written, it may even overshadow mainstream software development websites.

RQ5 Conclusion: Google finds software resources mostly on Stack Overflow (11%), YouTube (6%), and W3Schools (5%). However, the results may vary according to query (keyword or general).

RQ6: How do Google results vary for similar queries?

In this final analysis, we explore how Google results vary for similar developers' queries. The results are summarized in Table XV. Column "full agreement" presents the percentage of original and modified search query results with the same links and same order. The last column presents the intersection of links between the original and modified search query results. Overall, we observe that the top 1 results are less impacted by the modifications, that is, they often present high full agreement. For example, swapping the term *how to* from the beginning to end of the query does not change 80% of the top 1 results. On the other hand, notice that the top 10 results rarely have full agreement, at most 5%. However, with a few exceptions, the intersection of links is high between the original and modified results, over 70% in most of the comparisons. Next, we detail each category.

TABLE XV
AGREEMENT IN THE GOOGLE SEARCH RESULTS.

Cat.	Modification Type	Full agreement (%) Top 1	Full agreement (%) Top 10	Intersection Top 10 (Med.)
Word swap	context	89	5	90
	how to	80	1	80
	what is	72	0	80
	example	73	0	80
	vs	88	3	90
	convert to	14	0	30
Word removal	context	32	0	20
	how to	77	2	90
	what is	56	0	70
	example	23	0	30
	vs	44	0	60
	convert to	71	0	70
	stop words	65	1	80
Synonymous word	python → py	70	0	70
	javascript → js	80	2	80
	java → jdk	47	0	60
	php → php5	30	0	20
	css → cascading style sheets	52	0	50
	get → obtain	70	1	70
	create → build	60	1	60
	remove → delete	76	1	70
	check → verify	69	2	70
	convert → transform	63	2	70
Similar query	python	65	4	80
	javascript	60	1	70
	java	60	5	80
	php	58	2	80
	css	61	5	80

Word swap. Overall, swapping the word in the query (*e.g.,* from "*example xml file*" to "*xml file example*") is not likely to impact the top 1 search result, however, it affects the links and order of the top 10 results. In this case, 5 out of 6 comparisons have top 1 agreement over 70%, meaning that the original and modified queries mostly lead to the same top 1 link. The sole exception is the *convert to* queries: the agreement is low in the top 1 (14%) and the intersection of links is only 30%. In this case, the modified query may change the meaning of the original query. For example, the queries "*convert txt to json*" and "*convert json to txt"* have different goals, thus, it is expected they return distinct links.

Word removal. Omitting words from the queries impact the top 1 as well as the top 10 links and order. In the top 1, the agreement varies from 23 to 77%, while in the top 10 it varies from 0 to 2%. The least affected are the *how to* queries: after removing this term, 77% of the top 1 results are still the same. In contrast, the most affected are *example* and *context* queries, which have only 23% and 32% of agreement in the

top 1, respectively. Notice that, removing those terms may make the query generic: for example, removing *php* from "*php email parser*" to "*email parser*" causes it to be technology independent. Lastly, omitting stop words from queries affects 35% of the top 1 results, *i.e.,* 65% remains the same. However, the links have a high intersection (80%).

Synonymous word. For synonymous languages, the top 1 agreement has a large variation, from 30% to 80%. JavaScript and Python queries are the least affected: using *javascript* or *js* in the search queries is likely to return the same top 1 link (80% of agreement), with 80% of intersection; likewise, *python* and *py* produce 70% for both metrics. The most impacted results happens when we replace *php* by *php5* and *java* by *jdk*, leading to top 1 agreement of only 30% and 47%, respectively. Replacing *css* by *cascading style sheets* modifies around 50% of the top 1 results and the link intersection. Finally, synonymous verbs have less impact: the top 1 agreement ranges from 60% (create → build) to 76% (remove → delete), while the intersection is mostly around 70%.

Similar query. In this category, we rely on the data computed for RQ4, that is, Jaccard similarity and threshold 0.8. This way, we can compare the results of queries like "*change color svg css"* and *"svg change color in css"*. In this case, the top 1 agreement varies from 58% to 65%, while the top 10 from 1% to 5%. The intersection of links is high, mostly 80%.

RQ6 Conclusion: The links and order of the top 10 Google search results are very likely to change due to similar queries, whereas the top 1 is much less affected. However, overall, the intersection of links due to similar queries is high, at least 70% in most cases.

V. Discussion and Implications

Characteristics of developers' queries (RQ1, 2, 3 & 4). We find that developers' queries are likely to include key contexts. Most queries are similar among each other, that is, 60% have at least one similar peer. Moreover, we detect that developers' queries share some characteristics with general ones [31], [32], [36]: they are both short (3 words) and tend to omit function words. We also quantified several common terms, *e.g.,* 5w1h, comparison, and conversion. Thus, we contribute to the literature with a large study to characterize real-world developers' queries, complementing existing research on this field [5], [14], [16]–[19]. We reveal novel empirical data that can guide software vendors and content providers better understanding developers' queries and needs.

Patterns in developer's queries (RQ3). A common query starts with a technology and is followed by a verb and a complement, like <*technology*> <*verb*> <*complement*>, *e.g.,* "*python get temp dir*". We identify others patterns, like <*data1*> *vs* <*data2*> and *convert* <*data1*> *to* <*data2*>. Exploring those patterns can be useful for research areas as automated suggestions and reformulation of search queries [21]–[23]. For this purpose, queries can be suggested/refined according to the prevalence of certain verbs and nouns typically found in developers' queries. For example, developers often convert data types like string, array, and json, so conversion queries could be suggested based on this list of known nouns. Thus, we shed light on the existence of query patterns specific to software development. However, although we provide initial insights, a deeper analysis should yet be done.

Impact of changes on developers' queries (RQ6). Performing minor changes to queries (*e.g.,* swapping and omitting words) do not broadly affect the top 1 search results nor the overall top 10. Understating which changes can (and cannot) be safely applied to the queries have implications to tool creators that rely on web search to find errors/exceptions [24] and to integrate IDEs [25]–[27]. For example, tool creators can refine their queries (*e.g.,* when searching for similar error messages, code examples, etc.) and evaluate the impact on the users. Thus, our empirical data on the overall stability of Google results can foment the creation of better programming tools that rely on web search.

Cosmetic changes (RQ6). We have highlighted that the Google search results are not likely to change due to minor changes, however, we do find a few exceptions that deserve some attention. For example, one may argue that the simple swapping of the language from the beginning to the end of the query should ideally not affect the search results (*e.g.,* "*php utc time*" → "*utc time php*"), however, it does change 11% of the top 1 results and 10% of the top 10 links. Likewise, starting or ending the query with *how to* impacts 20% of the top 1 results and 20% of the top 10 links. We thus show that cosmetic query changes can affect the Google results, indicating that there is some room for improvement in the search algorithm, particularly in the software resource search landscape.

Omission of function words (RQ3 & 6). Users tend to omit function words in general search queries [32], [36]. We find that developers also tend to exclude function words from queries, which are mostly formed by content words (80.3%). However, we also detect that removing stop words (which includes function words) affect 35% of the top 1 search results and 20% of the returned links. This way, although developers may omit function words from the queries, they should be aware that the search results are sensitive to those changes. Also, tools that rely on web search (*e.g.,* [24]–[27]) may be impacted by those omissions and thus should be aware to avoid noisy results for developers.

Priority of Google to Stack Overflow (RQ5) It is fact that Stack Overflow plays an important role in software development nowadays [9], [47], [48]. Our findings reinforce this statement: we detect that Google finds software resources mostly on Stack Overflow (11%) with an over-concentration in the top 1 results (28%). Despite its popularity, recent studies warn that Stack Overflow should be used with caution: their code examples may be insecure, include outdated code, and contain usage and license violations [49]–[52]. The fact that Stack Overflow is mostly top 1 deepens this issue since those links receive more clicks from the users [53], consequently, the potential risks are even more widespread. We thus bring to light novel data about the priority Google search provides

to Stack Overflow, despite its well-known risks [49]–[52].

YouTube and other relevant sources (RQ5). YouTube is likely to be in the top 3 results of general queries, *e.g.,* the ones with *how to*, *what is*, etc. Interestingly, recent [9] and older [46] studies to assess resource search on the web do *not* point YouTube as a relevant source; this may happen because their search queries are specific (*i.e.,* API documentation). In contrast, we show that YouTube is a prominent source for Google. However, YouTube links are more likely to be returned in general queries than in specific ones. This can foment more research linking YouTube and software development (*e.g.,* [54]–[56]) to better understand how, why, and when developers resort to videos rather than programming websites.

Besides, we also present that Google finds software resources on a large variety of websites (we found 7K websites and 49K links for 5K queries in RQ5, and make it publicly available). This way, we provide novel resources for crowdsourcing studies [57], *e.g.,* to improve documentation [58]–[60], APIs [61]–[64], and code examples [41], [65], [66].

VI. Threats to Validity

Developers' queries and Alexa. Our queries come from Alexa, a platform that monitors global internet traffic [15]. Despite it is a major player on the market, their traffic is not complete; no tool can provide the whole internet traffic. Thus, despite our large analysis of 1.3M queries, they may represent a fraction of the whole traffic of the selected websites. Moreover, we do not differentiate the queries according to their frequency (*e.g.,* some queries may be more popular than others), thus, further studies should be performed to assess query frequency.

Manual classification of developers' intent. We rely on the categories proposed by Xia *et al.* [5] to assess developers' intention, keeping consistency with prior literature. Since the categories are well-defined, the query classification is straightforward. Even so, like any other manual classification, it is subjected to error and bias. However, an evidence that may minimize this threat is that both results share similarities (*e.g.,* reuse is common and testing is rare in both studies).

Google search results. We rely on the official Google Search API [20] to programmatically search on Google. This avoids any bias caused by manual search analysis (*e.g.,* cache, location, etc.) and allows us to scale to run thousands of queries. Notice that the Google Search API has request limits per day (10K), thus we could not scale to run millions of queries.

Personally identifiable information (PII). Alexa only keeps track of frequent queries, thus, they are unlikely to contain PII. We inspected 384 randomly selected queries (95% confidence level and 5% confidence interval); none of them included PII.

Generalization. The studied queries come from quite popular websites for developers and represent real-world needs. Also, we run the search queries on Google, which dominates the web with more than 92% of the search market share [45]. Despite this, as usual in empirical studies, our query analysis (RQ1-RQ4) cannot be directly generalized to other websites. Similarly, our result analysis (RQ5-RQ6) cannot be generalized to other search engines, like Yahoo!, Bing, Baidu, etc.

VII. Related Work

There exist a large research field on code search engines [8], [12], [41], [42], [67], [68]. In this context, some studies have been done to assess how developers search for code. Bajracharya *et al.* [16], [17] mined the log usage of the Koders code search engine and reveled some similarities with general web search, the prevalence of certain topics, and lexical structures. Sadowski *et al.* [18] performed a case study at Google to learn how developers search for code, including search frequency, search target, and developers' goal.

Xia *et al.* [5] studied the queries from 60 developers and surveyed 235 software engineers to understand developers' search tasks. They found that the most frequent search tasks are related to debugging, code reuse, and general searches, whereas we found that developers mostly perform code reuse and general searches. While the authors analyze queries of developers who work on outsourcing companies mostly in Java, in our study, the queries are broader and not specific to any language. Also, their goal is to understand the developers' search tasks, whereas we focus on developers' search queries themselves. Besides, we analyze the search results due to developers' queries with the Google Search API [20], which is a novel assessment in the literature. In a related line, Microsoft researchers [19] provided a classifier for distinguishing software engineering related queries from other queries and defined the taxonomy of queries. Rahman *et al.* [14] focused on how developers use general web search for code retrieval by assessing the queries of 310 developers. The authors compare code and non-code related queries regarding vocabulary, modification, and sessions, and find that code search performance in general web search is less effective. In this study, we do not contrast code and non-code queries. We assess other aspects of developers' search queries: content, size, keyword position, structure, similarity, result resources, and result variation. Thus, we complement existing studies with novel data about developers' queries and search results.

VIII. Conclusion

We presented a large empirical study to understand what developers search for on the web and what they find. We first performed a quantitative and qualitative analysis to assess the content, size, structure, and similarity of 1.3M developers' queries. Then, we run thousands of search queries on Google to explore the search results. We found that developers' queries typically include references to key contexts, are short, and tend to omit functional words; minor changes to queries do not largely affect the search results; and top search results are dominated by Stack Overflow and YouTube. Lastly, we discussed and provided implications for developers and researchers.

As future work, we plan to deepen our analysis on the query patterns specific to development (*e.g.,* comparison, conversion, etc.). We also plan to perform more qualitative analysis per context to better understand specific needs. Lastly, we plan to expand the set of modifications to the search queries to assess its impact and implications on the Google search results.

REFERENCES

[1] X. Gu, H. Zhang, D. Zhang, and S. Kim, "Deep API learning," in *ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 631–642, https://doi.org/10.1145/2950290.2950334.

[2] K. T. Stolee, S. Elbaum, and D. Dobos, "Solving the search for source code," *ACM Transactions on Software Engineering and Methodology*, vol. 23, no. 3, p. 26, 2014, https://doi.org/10.1145/2581377.

[3] S. E. Sim, M. Umarji, S. Ratanotayanon, and C. V. Lopes, "How well do search engines support code retrieval on the web?" *ACM Transactions on Software Engineering and Methodology*, vol. 21, no. 1, p. 4, 2011, https://doi.org/10.1145/2063239.2063243.

[4] S. E. Sim, M. Agarwala, and M. Umarji, "A controlled experiment on the process used by developers during internet-scale code search," in *Finding Source Code on the Web for Remix and Reuse*. Springer, 2013, pp. 53–77, https://doi.org/10.1007/978-1-4614-6596-6_4.

[5] X. Xia, L. Bao, D. Lo, P. S. Kochhar, A. E. Hassan, and Z. Xing, "What do developers search for on the web?" *Empirical Software Engineering*, vol. 22, no. 6, pp. 3149–3185, 2017, https://doi.org/10.1007/s10664-017-9514-4.

[6] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code," in *Conference on Human Factors in Computing Systems*, 2009, pp. 1589–1598, https://doi.org/10.1145/1518701.1518944.

[7] K. Philip, M. Umarji, M. Agarwala, S. E. Sim, R. Gallardo-Valencia, C. V. Lopes, and S. Ratanotayanon, "Software reuse through methodical component reuse and amethodical snippet remixing," in *Conference on Computer Supported Cooperative Work*, 2012, pp. 1361–1370, https://doi.org/10.1145/2145204.2145407.

[8] H. Niu, I. Keivanloo, and Y. Zou, "Learning to rank code examples for code search engines," *Empirical Software Engineering*, vol. 22, no. 1, pp. 259–291, 2017, https://doi.org/10.1007/s10664-015-9421-5.

[9] C. Treude and M. Aniche, "Where does Google find API documentation?" in *International Workshop on API Usage and Evolution*, 2018, pp. 19–22, https://doi.org/10.1145/3194793.3194796.

[10] Stack Overflow, https://stackoverflow.com, July, 2020.

[11] W3Schools, https://www.w3schools.com, July, 2020.

[12] J. Kim, S. Lee, S.-w. Hwang, and S. Kim, "Towards an intelligent code search engine," in *Conference on Artificial Intelligence*, 2010, p. 1358–1363, https://dl.acm.org/doi/10.5555/2898607.2898824.

[13] M. Raghothaman, Y. Wei, and Y. Hamadi, "Swim: Synthesizing what I mean-code search and idiomatic snippet synthesis," in *International Conference on Software Engineering*, 2016, pp. 357–367, https://doi.org/10.1145/2884781.2884808.

[14] M. M. Rahman, J. Barson, S. Paul, J. Kayani, F. A. Lois, S. F. Quezada, C. Parnin, K. T. Stolee, and B. Ray, "Evaluating how developers use general-purpose web-search for code retrieval," in *International Conference on Mining Software Repositories*, 2018, pp. 465–475, https://doi.org/10.1145/3196398.3196425.

[15] Alexa, https://www.alexa.com, July, 2020.

[16] S. Bajracharya and C. Lopes, "Mining search topics from a code search engine usage log," in *International Working Conference on Mining Software Repositories*, 2009, pp. 111–120, https://doi.org/10.1109/MSR.2009.5069489.

[17] S. K. Bajracharya and C. V. Lopes, "Analyzing and mining a code search engine usage log," *Empirical Software Engineering*, vol. 17, no. 4-5, pp. 424–466, 2012, https://doi.org/10.1007/s10664-010-9144-6.

[18] C. Sadowski, K. T. Stolee, and S. Elbaum, "How developers search for code: a case study," in *European Software Engineering Conference and the Symposium on the Foundations of Software Engineering*, 2015, pp. 191–201, https://doi.org/10.1145/2786805.2786855.

[19] C. Bansal, T. Zimmermann, A. H. Awadallah, and N. Nagappan, "The usage of web search for software engineering," *arXiv preprint arXiv:1912.09519*, 2019, https://arxiv.org/pdf/1912.09519.pdf.

[20] Google Search API, https://developers.google.com/custom-search/v1/overview, July, 2020.

[21] S. Haiduc, G. Bavota, A. Marcus, R. Oliveto, A. De Lucia, and T. Menzies, "Automatic query reformulations for text retrieval in Software Engineering," in *International Conference on Software Engineering*, 2013, pp. 842–851, https://doi.org/10.1109/ICSE.2013.6606630.

[22] M. M. Rahman and C. Roy, "Effective reformulation of query for code search using crowdsourced knowledge and extra-large data analytics," in *International Conference on Software Maintenance and Evolution*, 2018, pp. 473–484, https://doi.org/10.1109/ICSME.2018.00057.

[23] M. M. Rahman, C. K. Roy, and D. Lo, "Automatic query reformulation for code search using crowdsourced knowledge," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1869–1924, 2019, https://doi.org/10.1007/s10664-018-9671-0.

[24] M. M. Rahman, S. Yeasmin, and C. K. Roy, "Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions," in *Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering*, 2014, pp. 194–203, https://doi.org/10.1109/CSMR-WCRE.2014.6747170.

[25] J. Brandt, M. Dontcheva, M. Weskamp, and S. R. Klemmer, "Example-centric Programming: Integrating Web Search into the Development Environment," in *SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 513–522, https://doi.org/10.1145/1753326.1753402.

[26] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, and M. Lanza, "Mining StackOverflow to turn the IDE into a self-confident programming prompter," in *Working Conference on Mining Software Repositories*, 2014, pp. 102–111, https://doi.org/10.1145/2597073.2597077.

[27] M. M. Rahman, C. K. Roy, and D. Lo, "Rack: Code search in the ide using crowdsourced knowledge," in *International Conference on Software Engineering Companion*, 2017, pp. 51–54, https://doi.org/10.1109/ICSE-C.2017.11.

[28] GeeksForGeeks, https://www.geeksforgeeks.org, July, 2020.

[29] TutorialsPoint, https://www.tutorialspoint.com, July, 2020.

[30] ProgramCreek, https://www.programcreek.com, July, 2020.

[31] A. Spink, B. J. Jansen, D. Wolfram, and T. Saracevic, "From e-sex to e-commerce: Web search changes," *Computer*, vol. 35, no. 3, pp. 107–109, 2002, https://doi.org/10.1109/2.989940.

[32] A. J. Biega, J. Schmidt, and R. S. Roy, "Towards query logs for privacy studies: On deriving search queries from questions," in *European Conference on Information Retrieval*, 2020, pp. 110–117, https://doi.org/10.1007/978-3-030-45442-5_14.

[33] M. Kestemont, "Function words in authorship attribution. from black magic to theory?" in *Workshop on Computational Linguistics for Literature*, 2014, pp. 59–66, http://doi.org/10.3115/v1/W14-0908.

[34] C. Chung and J. W. Pennebaker, "The psychological functions of function words," *Social communication*, vol. 1, pp. 343–359, 2007, https://psycnet.apa.org/record/2007-01308-012.

[35] G. K. Zipf, *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books, 2016, https://doi.org/10.2307/2572028.

[36] C. Barr, R. Jones, and M. Regelson, "The linguistic structure of english web-search queries," in *Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 1021–1030, https://dl.acm.org/doi/10.5555/1613715.1613848.

[37] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975, https://doi.org/10.1145/361219.361220.

[38] P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901, https://ci.nii.ac.jp/naid/10027880482.

[39] F. Chierichetti, R. Kumar, S. Pandey, and S. Vassilvitskii, "Finding the Jaccard median," in *Symposium on Discrete Algorithms*, 2010, pp. 293–311, https://doi.org/10.1137/1.9781611973075.25.

[40] D. Silva and M. T. Valente, "RefDiff: detecting refactorings in version histories," in *International Conference on Mining Software Repositories*, 2017, pp. 269–279, https://doi.org/10.1109/MSR.2017.14.

[41] I. Keivanloo, J. Rilling, and Y. Zou, "Spotting working code examples," in *International Conference on Software Engineering*, 2014, pp. 664–675, https://doi.org/10.1145/2568225.2568292.

[42] X. Gu, H. Zhang, and S. Kim, "Deep code search," in *International Conference on Software Engineering*, 2018, pp. 933–944, https://doi.org/10.1145/3180155.3180167.

[43] C. D. Manning, H. Schütze, and P. Raghavan, *Introduction to information retrieval*. Cambridge university press, 2008, https://nlp.stanford.edu/IR-book.

[44] Google Custom Search Engine, https://developers.google.com/custom-search/docs/tutorial/creatingcse, July, 2020.

[45] Search Engine Market Share Worldwide, https://gs.statcounter.com/search-engine-market-share, July, 2020.

[46] C. Parnin and C. Treude, "Measuring API documentation on the web," in *International Workshop on Web 2.0 for Software Engineering*, 2011, pp. 25–30, https://doi.org/10.1145/1984701.1984706.

[47] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow," *Georgia Institute of Technology, Tech. Rep*, 2012, http://chrisparnin.me/pdf/crowddoc.pdf.

[48] State of the Stack 2019: A Year in Review, https://stackoverflow.blog/2019/01/18/state-of-the-stack-2019-a-year-in-review, July, 2020.

[49] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack Overflow considered harmful? the impact of copy&paste on Android application security," in *Symposium on Security and Privacy*, 2017, pp. 121–136, https://doi.org/10.1109/SP.2017.31.

[50] N. Meng, S. Nagy, D. Yao, W. Zhuang, and G. A. Argoty, "Secure coding practices in Java: Challenges and vulnerabilities," in *International Conference on Software Engineering*, 2018, pp. 372–383, https://doi.org/10.1145/3180155.3180201.

[51] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online Q&A forum reliable?: a study of API misuse on Stack Overflow," in *International Conference on Software Engineering*, 2018, pp. 886–896, https://doi.org/10.1145/3180155.3180260.

[52] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, "Toxic code snippets on Stack Overflow," *Transactions on Software Engineering*, 2019, https://doi.org/10.1109/TSE.2019.2900307.

[53] Google General Guidelines, https://static.googleusercontent.com/media/www.google.com/en//insidesearch/howsearchworks/assets/searchqualityevaluatorguidelines.pdf, July, 2020.

[54] L. MacLeod, M.-A. Storey, and A. Bergen, "Code, camera, action: How software developers document and share program knowledge using YouTube," in *International Conference on Program Comprehension*, 2015, pp. 104–114, https://doi.org/10.1109/ICPC.2015.19.

[55] L. Ponzanelli, G. Bavota, A. Mocci, M. Di Penta, R. Oliveto, M. Hasan, B. Russo, S. Haiduc, and M. Lanza, "Too long; didn't watch! extracting relevant fragments from software development video tutorials," in *International Conference on Software Engineering*, 2016, pp. 261–272, https://doi.org/10.1145/2884781.2884824.

[56] M. Ellmann, A. Oeser, D. Fucci, and W. Maalej, "Find, understand, and extend development screencasts on YouTube," in *ACM SIGSOFT International Workshop on Software Analytics*, 2017, pp. 1–7, https://doi.org/10.1145/3121257.3121260.

[57] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and GitHub: Associations between software development and crowdsourced knowledge," in *International Conference on Social Computing*, 2013, pp. 188–195, https://doi.org/10.1109/SocialCom.2013.35.

[58] M. M. Rahman, C. K. Roy, and I. Keivanloo, "Recommending insightful comments for source code using crowdsourced knowledge," in *International Working Conference on Source Code Analysis and Manipulation*, 2015, pp. 81–90, https://doi.org/10.1109/SCAM.2015.7335404.

[59] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from Stack Overflow," in *International Conference on Software Engineering*, 2016, pp. 392–403, https://doi.org/10.1145/2884781.2884800.

[60] M. P. Robillard, A. Marcus, C. Treude, G. Bavota, O. Chaparro, N. Ernst, M. A. Gerosa, M. Godfrey, M. Lanza, M. Linares-Vásquez *et al.*, "On-demand developer documentation," in *International Conference on Software Maintenance and Evolution*, 2017, pp. 479–483, https://doi.org/10.1109/ICSME.2017.17.

[61] M. M. Rahman, C. K. Roy, and D. Lo, "Rack: Automatic API recommendation using crowdsourced knowledge," in *International Conference on Software Analysis, Evolution, and Reengineering*, vol. 1, 2016, pp. 349–359, https://doi.org/10.1109/SANER.2016.80.

[62] S. Nadi, S. Krüger, M. Mezini, and E. Bodden, "Jumping through hoops: Why do java developers struggle with cryptography apis?" in *International Conference on Software Engineering*, 2016, pp. 935–946, https://doi.org/10.1145/2884781.2884790.

[63] J. Zhang, H. Jiang, Z. Ren, and X. Chen, "Recommending apis for api related questions in stack overflow," *IEEE Access*, vol. 6, pp. 6205–6219, 2017, https://doi.org/10.1109/ACCESS.2017.2777845.

[64] H. Jiang, J. Zhang, Z. Ren, and T. Zhang, "An unsupervised approach for discovering relevant tutorial fragments for apis," in *International Conference on Software Engineering*, 2017, pp. 38–48, https://doi.org/10.1109/ICSE.2017.12.

[65] R. P. Buse and W. Weimer, "Synthesizing API usage examples," in *International Conference on Software Engineering*, 2012, pp. 782–792, https://doi.org/10.1109/ICSE.2012.6227140.

[66] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, and A. Marcus, "How can i use this method?" in *International Conference on Software Engineering*, 2015, pp. 880–890, https://doi.org/10.1109/ICSE.2015.98.

[67] S. Bajracharya, T. Ngo, E. Linstead, Y. Dou, P. Rigor, P. Baldi, and C. Lopes, "Sourcerer: a search engine for open source code supporting structure-based search," in *Symposium on Object-oriented Programming Systems, Languages, and Applications*, 2006, pp. 681–682, https://doi.org/10.1145/1176617.1176671.

[68] D. Rush and A. Bulsara, "Source Code Search Engine," Dec. 27 2007, uS Patent App. 11/663,417, https://patents.google.com/patent/US20070299825A1/en.