Understanding How and Why Developers Seek and Analyze API-related Opinions

Gias Uddin, Olga Baysal, Latifa Guerrouj, and Foutse Khomh

Abstract—With the advent and proliferation of online developer forums as informal documentation, developers often share their opinions about the APIs they use. Thus, opinions of others often shape the developer's perception and decisions related to software development. For example, the choice of an API or how to *reuse* the functionality the API offers are, to a considerable degree, conditioned upon what other developers think about the API. While many developers refer to and rely on such opinion-rich information about APIs, we found little research that investigates the use and benefits of public opinions. To understand how developers seek and evaluate API opinions, we conducted two surveys involving a total of 178 software developers. We analyzed the data in two dimensions, each corresponding to specific needs related to API reviews: (1) Needs for seeking API reviews, and (2) Needs for automated tool support to assess the reviews. We observed that developers seek API reviews and often have to summarize those for diverse development needs (e.g., API suitability). Developers also make conscious efforts to judge the trustworthiness of the provided opinions and believe that automated tool support for API reviews analysis can assist in diverse development scenarios, including, for example, saving time in API selection as well as making informed decisions on a particular API features.

Index Terms—Opinion mining; API informal documentation; opinion summaries; survey; opinion quality; developer's perception.

1 Introduction

APIs (Application Programming Interfaces) offer interfaces to reusable software components. Modern-day rapid software development is often facilitated by the plethora of open-source APIs available for any given development task. The online development portal GitHub [1] now hosts more than 67 million public repositories. We can observe a radical increase from the 2.2 million active repositories hosted in GitHub in 2014. While many of the public repositories in GitHub may not be code-based or are personal projects, we observed similar growth in many other online API package managers. For example, we observed an increase in the number of open-source APIs shared in all the following package managers (as of July 2018) : 1) 11,782% for Javascript APIs in *npm* package manager (from 5,646 in December 2011), 2) 334% for Java APIs in online *maven* central (from 55,785 in March 2013), 3) 2,447% for C# APIs in online *nuget* repository (from 4,799 in February 2012), 4) 1,477% for Python APIs in *PyPI* (from 9,362 in March 2010). Javascript, Java, C# and Python are among the top five most popular programming languages in Stack Overflow[1]. Developers can share their APIs in other package managers as well, such as *Bower* for Javascript, *Rubygems.org* for Ruby, etc.

With a myriad of APIs being available, developers now face a new challenge — how to choose the right API. For any given task, we now expect to see multiple competing APIs. For example, in the mapping community, developers can choose from multiple web APIs, such as Google Maps APIs, Bing Maps APIs, Apple Maps APIs, MapBox, OpenLayer, etc. The selection and adoption of an API depends on a number of factors [2], such as the availability of learning resources, the design and usability of the API [3], [4], [5]. Developers can learn APIs by using the API official documentation. However, the official documentation can be often incomplete, obsolete, and/or incorrect [6], [7]. In our previous study of more than 300 developers at IBM, we found that such problems in an API official documentation can motivate a developer to select other competing APIs.

To overcome the challenge of selecting an API among available choices and properly learning it, many developers seek help and insights from other developers in online developer forums. Figure 1 presents the screenshot of seven Stack Overflow posts. In Figure 1, the four red-coloured circles are answers ①, ③-⑤ and three green-coloured circles are comments ②, ⑥, ⑦. The oldest post (at the top) is dated from November 06, 2009, while the most recent one (at the bottom) is from February 10, 2016. These posts express developers' opinions about two Java APIs (Jackson [8] and Gson [9]) offering JSON parsing features for Java. None of the posts contain any code snippets. The first answer ① representing a positive opinion about the Gson API motivates the developer 'binaryrespawn' to use it ②. In the next answer ③, the user 'StaxMan' compares Gson with Jackson, favoring Jackson for offering better support, and based on this feedback, 'mickthomson' ④ decides to use Jackson instead of Gson. Three out of the four answers ③–⑤ imply a positive sentiment towards Jackson but a negative one about Gson. Later, the developer 'Daniel Winterstein' develops a new version of Gson fixing existing issues, and shares his API ⑦. This example illustrates how developers share their experiences and insights, as well as how they influence and are influenced by other developers' opinions. A developer looking for only code examples for Gson would have missed the important insights about the API's limitations, which may have affected his development activities. Thus, opinions extracted from the informal discussions can drive developers' decision making.

Indeed, opinions are key determinants to many of the activities related to software development, such as developers' productivity analysis [10], determining developers burnout [11], improving software applications [12], and developing awareness tools for software development teams [13], [14], [15]. Research on APIs has produced important contributions, such as automatic usage inference mining [16], [17], automatic traceability recovery between

1. We used the GitHub API and modulecounts.com to collect the statistics

Fig. 1: Example of a Stack Overflow discussion about two APIs. Green-coloured circles are comments, and red-coloured ones are answers.

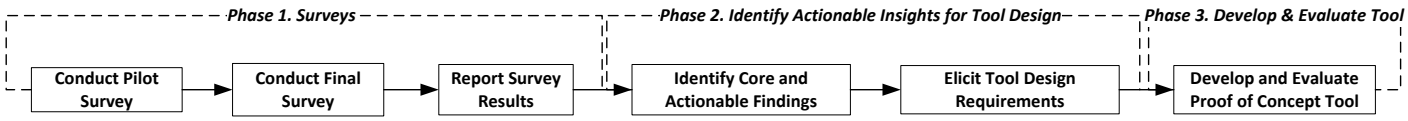

Fig. 2: The major research steps undertaken during and after the two surveys reported in this paper

API elements and learning resources [18], [19], [20], as well as recommendation systems to facilitate code reuse [21], [22] (see Section 2). However, to the best of our knowledge, there is no research that focuses on the analysis of developers' perception about API reviews and how such opinions affect their API-related decisions. As illustrated in Figure 1, while developer forums serve as communication channels for discussing the implementation of the API features, they also enable the exchange of opinions or sentiments expressed on numerous APIs, their features and aspects. Given the presence of sentiments in the forum posts and opinions about APIs, such insights can be leveraged to develop techniques to automatically analyze API reviews in forum posts. Such insights can also contribute not only to the development of an empirical body of knowledge on the topic, but also to the design of tools that analyze opinion-rich information.

To fill this gap in the literature, we conducted two surveys involving a total of 178 software developers. The *goals* of our study are to *understand* (1) how software developers seek and value opinions about APIs, and (2) what tools can better support their analysis and evaluation of the API reviews. The *subjects* are the surveys' participants and the *objects* are the API reviews that the developers encounter in their daily development activities from diverse resources. The *context* consists of the various development activities that can be influenced by the API reviews. Through an exploratory analysis of the surveys' responses, we answer the following research questions:

RQ1: How do developers seek and value opinions about APIs in developer forums?

The developers reported that they seek opinions about APIs in forum posts to support diverse development needs, such as

API selection, documentation, learning how to use an API, etc. The developers valued the API reviews. However, they were also cautious while making informed decisions based on those reviews due to a number of factors related to the quality of the provided opinions, such as, the lack of insights into the prevalence of the issue reported in the opinion, the trustworthiness of the provided opinion (e.g., marketing initiatives vs subjective opinion), etc. The developers wished for the support of different mechanisms to aggregate opinions about APIs and to assess the quality of the provided opinions about APIs in the developer forums.

RQ2: What tool support is needed to help developers with analyzing and assessing API reviews in developer forums?

The developers consider that automated tools of diverse nature can be developed and consulted to properly analyze API reviews in forum posts, e.g., visualizations of the aggregated sentiment about APIs to determine their popularity, understanding the various aspects (e.g., performance) about APIs and how other developers rate those aspects about APIs based on their usage of the APIs, etc. The developers mentioned that the huge volume of available information and opinions about APIs in the forum posts can hinder their desire to get quick, digestible, and actionable insights about APIs. The developers also mentioned that in the absence of any automated summarization technique to help them, they leverage different features in forum posts to get summarized viewpoints of the APIs, e.g., skimming through highly-ranked posts, using tags to find similar APIs, etc. Developers also envision diverse potential summarization approaches to help them address such needs, e.g., a dedicated portal to show aggregated opinions about APIs.

In Figure 2, we show the three major research phases we undertook during and after conducting the two surveys.

Phase 1. Design, Conduct, and Report Surveys
We conducted two surveys. We analyze the survey responses using both statistical and qualitative analyses. This paper primarily focuses on the design, analysis, and reporting of the two surveys (see Sections 3 - 7).

Phase 2. Identify Actionable Insights for Tool Design
We identify requirements from the survey results to develop techniques and tools to assist developers in their exploration of opinions about APIs from developer forums. In Section 8, we discuss the actionable findings from the survey results that could be used for future tool designs.

Phase 3. Develop and Evaluate Techniques and Tools
We develop techniques based on the findings from Phase 2. We incorporate the techniques in our prototype tool, called Opiner. Opiner is a search engine [23], [23]. Using Opiner, developers can search for an API by name and explore the opinions and usage scenarios related to APIs. The opinions and usage scenarios are automatically mined from Stack Overflow. In Section 8, we briefly describe Opiner.

In this paper, we make the following main contributions:

1) **Surveys.** The design of two surveys and the collected data involving the responses of 178 software engineers.
2) **Analysis.** A detailed analysis of the survey responses that provides insights into: (1) How developers seek and analyze opinion-rich API information. (2) Needs for automated tool supports to make informed, proactive, and efficient decisions based on analysis of API reviews. In Table 1, we compare the major findings of this paper against the state of the art research on APIs. In Section 2, we discuss the related work in details.

2 Related Work

As noted in Section 1, the findings of this paper motivated us to pursue a research journey that contributed to the development of our proof-of-concept tool, Opiner (see Section 8). Specifically, our subsequent research projects focused on the design, development, and evaluation of techniques and tools to mine and summarize opinions and usage scenarios about APIs from developer forums. The research journey is captured in the following manuscripts:

1) In [50], we present a benchmark dataset of 4,522 sentences from Stack Overflow, each labelled as API aspects, such as performance, usability, etc. The catalog of API aspects is derived from the two surveys of this paper (Q11 and Q15 in Final Survey and Q15 from pilot survey). We leverage the benchmark dataset to develop machine learning supervised classifiers to automatically detect API aspects discussed in opinionated sentences. We then present a suite of algorithms to automatically mine opinions about APIs from Stack Overflow. We report the evaluation of each technique.
2) In [51], we present two algorithms to summarize opinions about APIs from Stack Overflow. The design and development of the algorithms were motivated by the findings from this paper, e.g., developers prefer to seek opinions about API aspects, such as performance, etc. We compare the algorithms against six off-the-shelf summarization algorithms.
3) In [23], we present the Opiner architecture that supports the mining and summarization of opinions from Stack Overflow.
4) In [52], we present a framework to automatically mine usage scenarios about APIs from Stack Overflow. We present an empirical study to investigate the value of the framework.
5) In [53], we present four algorithms to automatically summarize usage scenarios about APIs and their evaluation using four user studies.

As noted above, the findings from this paper have formed the cornerstone towards the development and evaluation of the techniques and tools presented in the above papers.

Other related work can be categorized into four areas. (1) Studies conducted to understand how developers learn to select and use APIs, (2) Analysis of APIs in developer forums, (3) Sentiment analysis in software engineering, and (4) Summarization of software artifacts. We discuss the related work below.

2.1 How developers learn to select and use APIs

While our surveys focused on the role of opinions to support development tasks, previous studies mainly focused on interviews or empirical studies to understand the role of API official documentation to support the development tasks [6], [7], [38], [54], [55], [56], [57]. Developers in our survey seek opinions about APIs to support diverse development needs (e.g., API selection) as well as to compensate for shortcomings in API documentation, e.g., when the documentation is incomplete or ambiguous. The problems in API official documentation are previously reported in multiple studies, such as [6], [7], [24], [58], etc.

Robillard and DeLine [7] conducted a survey and a series of qualitative interviews of software developers at Microsoft to understand how developers learn APIs. The study identified that the most severe obstacles developers faced while learning new APIs were related to the official documentation of the APIs. With API documentation, the developers cited the lack of code examples and the absence of task-oriented description of the API usage as some major blockers to use APIs. The benefit of task-based

TABLE 1: Comparison between our findings and prior findings

Theme	Our Study	Prior Study	Comparison
How developers learn to select and use APIs	(1) Developers seek opinions to support diverse development needs, e.g., API selection, feature improvement, etc. (2) Developers seek opinions to compensate for the shortcomings in API official documentation (e.g., incorrectness, etc.). (3) Developers consider the combination of code examples and opinions about an API as a form of API documentation.	Robillard and DeLine [7] identified in a series of survey and interviews that the most severe obstacles developers faced while learning new APIs were related to the official documentation of the APIs.	Our study confirms the findings of [7] that developers find API official documentation can be incomplete. In addition, we find that developers leverage API reviews in forums to compensate for those shortcomings.
		Carroll et al. [24] designed "minimal manual" to support task-based documentation after observing that the learning of the developers was often interrupted by their self-initiated problem-solving tasks while using API official documentation.	While Carroll et al. [24] created "minimal manual" manually, our results show that we can leverage developer forums to develop "minimal manual" by combining code examples and API reviews, because developers consider both as a form of documentation.
Sentiment analysis of software artifacts	(1) Developers use positive and negative opinions of other developers as an indicator of quality of the discussed API and code examples. (2) Developers face challenges to determine the quality of provided opinions (e.g., trustworthiness, recency, real-world facts, etc.).	The attributes in developer forums (e.g., upvotes, downvotes, etc.) are used to analyze the quality of posts and their roles in the Q&A process [25], [26], [27], [28], [29], [30], to analyze developer profiles (e.g., personality traits of the most and low reputed users) [31], [32], or to determine the influence of badges in Stack Overflow [33].	Our findings show that more insights can be derived by analyzing the opinions of developers towards APIs, in addition to [25], [26], [27], [28], [29], [30]. Our findings highlight the needs for research into the analysis of opinion quality, e.g., by gaining deeper insights into developer reputations and badges following [31], [32], [33].
Analysis of APIs in developer forums	Developers prefer to learn about different API aspects from the opinions of other developers in forum posts using automated analysis (e.g., find all opinions discussing about the performance of an API).	Zhang and Hou [34] identified problematic API features in the discussion Stack Overflow posts, by detecting sentences with negative sentiments. Treude and Robillard [35] mined important insights about an API type from the textual contents of Stack Overflow.	Unlike Zhang and Hou [34], our study shows that both positive and negative opinions about API features need to be identified. The insights gained for each API type by [35] can be enhanced by also including the diverse opinions about API aspects.
Summarization of software artifacts	Developers mostly rely on search engines to explore opinions about APIs. They are frequently overwhelmed with the huge volume of opinions about APIs in forums. They wished for an automatic summarizer by mining those opinions.	Topic modeling has been used to find dominant discussion topics in Stack Overflow [36], [37], and to find recurrent themes in API learning obstacles [38]. Both natural language and structural summaries of source code have been studied extensively [21], [39], [40], [41], [42], [43], [44], [45], [46].	Unlike [36], [37], our study shows that a finer grained topic-based summarization is required (e.g., using only opinions). Given the needs to analyze opinions by diverse API aspects (e.g., performance, usability), an aspect-based [47] opinion summarization for APIs can be useful, which is different from source code summarization [21], [39], [40], [41], [42], [43], [44], [45], [46].
	Besides an opinion summarizer, developers also asked for tool support to assist in their development tasks by leveraging opinions about APIs, such as, API comparator, trend analyzer, API opinion miner, etc.	Several tools have been developed to harness knowledge about APIs from developer forums, such as automatically generating comments to explain a code example [48], recommending experts to answer a question in Stack Overflow [49], etc.	Unlike [48], [49], we offer insights on the usage of opinions in tool development to support development tasks. Our findings offer possible extensions to existing research, e.g., include reactions towards a code example of [48] to show its quality.

documentation over traditional hierarchy-based documentation (e.g., Javadoc) was previously also reported by Carroll et al. [24] who observed developers while using traditional documentation to learn an API (e.g., Javadoc, manual). They found that the learning of the developers was often interrupted by their self-initiated problem-solving tasks that they undertook during their navigation of the documentation. During such unsupervised exploration, they observed that developers ignored groups and entire sections of a documentation that they deemed not necessary for their development task at hand. Unsurprisingly, such unsupervised exploration often led to mistakes. They conjectured that traditional API documentation is not designed to support such active way of developers' learning. To support developers' learning of APIs from documentation, they designed a new type of API documentation, called as the *minimal manual*, that is task-oriented and that helps the users resolve errors [24], [59], [60], [61]. In a subsequent study of 43 participants, Shull et al. [58] also confirmed the effectiveness of example-based documentation over hierarchy-based documentation.

The advent of cloud-based software development has popularized the adoption of Web APIs, such as, the Google Map APIs, API mashups in the ProgrammableWeb, and so on. Tian et al. [62] conducted an exploratory study to learn about the features offered by the Web APIs via the ProgrammableWeb portal and the type of contents supported in the documentation of those APIs. They observed that such Web APIs offer diverse development scenarios, e.g., text mining, business analysis, etc. They found that the documentation of the APIs offer insights into different knowledge types, e.g., the underlying intent of the API features, the step-by-step guides, etc. Sohan et al. [54] observed that REST API client developers face problems while using an API without usage examples, such as correct data types, formats, required HTTP headers, etc. Intuitively, the developer forums can be used for missing code examples for an API, because developers in our surveys mentioned that they rely on the API usage discussions in developer forums when the API official documentation can be missing.

The generation of such task-based documentation can be challenging. However, the developers in our survey reported that they consider the combinations of code examples and reactions towards the examples about an API in the forum as a form of API documentation. Intuitively, the Q&A format of online developer forums (e.g., Stack Overflow) follow task-based documentation format, e.g., the task is described in the question and the solution with code examples and opinions in the answer posts. Leveraging usage scenarios about APIs posted in the developer forums can be necessary, when API official documentation does not include those scenarios and can often be outdated [63]. Indeed,

documentation that does not meet the expectations of its readers can lead to frustration and a major loss of time, or even in an API being abandoned [7]. To address the shortcomings in API official documentation, research efforts focused on the linking of API types in a formal documentation (e.g., Javadoc) to code examples in forum posts where the types are discussed [64], presenting interesting textual contents from Stack Overflow about an API type in the formal documentation [65], etc. However, our study in this paper shows that the plethora of usage discussions available for an API in the forum posts can pose challenges to the developers to get quick and actionable insights into how an API can be used for a given development task. One possible way to assist developers is to generate on-demand developer documentation from forum posts [66]. However, to be able to do that, we first need to understand what specific problems persist in the API official documentation, that should be addressed through such documentation efforts. If we know of the common documentation problems, we can then prioritize those problems and investigate techniques to leverage API usage scenarios posted in the developer documents to address those problems. In a recent study [6], we conducted surveys of more than 300 developers at IBM to understand the problems developers faced while using API official documentation. We observed 10 common documentation problems, such as documentation incompleteness, incorrectness, ambiguities, etc. Therefore, we can leverage the findings from our study to design API documentation resources that can address the problems developers that are commonly observed in API documentation.

2.2 Sentiment Analysis of Software Artifacts

Developers in our surveys reported that they use the positive and negative opinions towards an API as an indicator of quality of the features offered by the API. While our findings shed light on developers' perceptions of API quality by leveraging opinions, recent research of sentiment analysis of software artifacts focused on the mining of sentiments and emotions from software repositories. Ortu et al. [10] observed weak correlation between the politeness of the developers in the comments and the time to fix the issue in Jira, i.e., bullies are not more productive than others in a software development team. Mäntylä et al. [11] correlated VAD (Valence, Arousal, Dominance) scores [67] in Jira issues with the loss of productivity and burn-out in software engineering teams. They found that the increases in issue's priority correlate with increases in Arousal. Pletea et al. [68] found that security-related discussions in GitHub contained more negative comments. Guzman et al. [69] found that GitHub projects written in Java have more negative comments as well as the comments posted on Monday, while the developers in a distributed team are more positive. Guzman and Bruegge [13] summarized emotions expressed across collaboration artifacts in a software team (bug reports, etc.) using LDA [70] and sentiment analysis. The team leads found the summaries to be useful, but less informative.

We observed in our surveys that while developers analyze opinions to learn about APIs, they also find it challenging to assess the quality (e.g., trustworthiness) of the provided opinions due to the presence of such opinions scattered across multiple unrelated forum posts. Related research has focused on the assessment of post quality using post attributes (e.g., post score) and their roles in the Q&A process [25], [26], [27], [28], [29], [30], to analyze developer profiles (e.g., personality traits of the most and low reputed users) [31], [32], or to determine the influence of badges in Stack Overflow [33]. In contrast to the above work, our findings motivate the needs to develop opinion quality assessment models, tools and techniques for forum posts.

2.3 Analysis of APIs in Developer Forums

Developers in our surveys prefer to seek opinions about different API aspects and wish to leverage automated tools to support such analysis (e.g., get all the opinions discussing about the performance of an API, etc.). A closely related work is the identification of problematic API features in Stack Overflow by Zhang and Hou [34], who considered sentences with negative sentences as indicators of problematic API features. In a parallel study, Wang and Godfrey [38] hypothesize that the more discussions an API class generate in the forum posts, the more likely that it is problematic to use. By applying topic modeling on the posts, they observed several recurrent themes of API usage obstacles, such as learning the interactions among components. Unlike both [34] and [38] our findings motivate the needs to study both positive and negative opinions about APIs to obtain finer-grained insights about API aspects. Intuitively, such fine-grained insights can also be useful to compare competing APIs for a given task.

A large volume of API research has devoted into the automatic mining of insights about APIs from Stack Overflow. In contrast to our study, they the studies are mainly empirical (i.e., automatic mining techniques) and they do not consider opinions about APIs. For example, Treude and Robillard [35] developed machine learning tools to detect insightful sentences about API types from Stack Overflow. Parnin et al. [71] have investigated API classes discussed in Stack Overflow using heuristics based on exact matching of classes names with words in posts (title, body, snippets, etc.). Using a similar approach, Kavaler et al. [30] analyzed the relationship between API usage and their related Stack Overflow discussions. Both studies found a positive relationship between API class usage and the volume of Stack Overflow discussions. More recent research [72], [73] investigated the relationship between API changes and developer discussions. Treude et al. [57] categorized questions and posts from Stack Overflow to understand the types of questions asked in Stack Overflow. They observed that developer forums are the most effective for code reviews and conceptual questions. The findings corroborate to our study participants who mentioned that they seek expertise in API usage in forum posts.

2.4 Summarization of Software Artifacts

Developers in our surveys wish for tools to automatically summarize the huge volume of opinions posted about APIs from developer forums. Such summaries can be different from the research conducted to produce Natural language and structural summaries of source code [21], [39], [40], [41], [42], [43], [44], [45], [46], [74], [75], [76].

One potential technique in text-based summarization is topic modeling, which can be applicable to summarize opinions (e.g., in other domains [47]). Previously, topic modeling has been used to study dominant discussion topics in developer forums [36], [37]. Our survey findings motivate for a finer grained analysis using topic modeling, such as analysis of only opinions instead of all textual contents. Given that developers in our survey prefer

to analyze opinions by API aspects (e.g., performance, usability), another potential summarization approach could be aspect-based summarization [47] of API opinions, as we developed in Opiner [51] based on the findings from this study.

3 Research Context

We further motivate the needs for better understanding of the impact of opinions about APIs on the developers first by taking cues from other domains (Section 3.1) and then by demonstrating the prevalence of opinions about APIs in Stack Overflow based on small study in Section 3.2. We discuss the rationale behind each research question, along with its sub-questions in Sections 3.3 and 3.4. We then discuss the motivation behind the two surveys we conducted to answer the research questions (Section 3.5).

3.1 Opinion Analysis in Other Domains

Our research on API reviews was motivated by similar research in the other domains. Automated sentiment analysis and opinion mining about entities (e.g., cars, camera products, hotels, restaurants) have been challenging but a practical research area due to their benefits for consumers (e.g., guiding them in choosing a hotel or selecting a camera product).

In Figure 3, we show the screenshot of three separate initiatives on the automatic collection and aggregation of reviews about camera products. The first two tools are developed in the academia and the third tool was developed as part of Microsoft's Bing Product Search. The first two circles ① and ②, show two preliminary outlines of such a tool presented by Liu et al. [77]. The third circle ③ shows a similar tool in the Microsoft Bing Product Search. In all of the three tools, positive and negative opinions about a camera product are collected and aggregated under a number of aspects (e.g., picture quality, battery life, etc.). When no predefined aspect is found, the opinions are categorized under a generic aspect "GENERAL" (aspect is named as "feature" in ②). The opinions about the camera can be collected from diverse sources, e.g., online product reviews, sites selling the product, etc. For example, Google collects reviews about hotels and restaurants through a separate gadget in its online search engine.

3.2 API Reviews in Developer Forums

Similar to the camera reviews in Figure 3, API reviews can be found in forum posts. As we demonstrated in Figure 1, opinions about APIs can be prevalent in developer forums. In fact, we observed that more than 66% of posts that are tagged "Java" and "JSON" in the Stack Overflow data contain at least one positive or negative sentiment[2]. In Table 2 we show descriptive statistics of the dataset. There were 22,733 posts from 3,048 threads with scores greater than zero. We did not consider any post with a negative score because such posts are considered as not helpful by the developers in Stack Overflow. The last column "Users" show the total number of distinct users that posted at least one answer/comment/question in those threads. To identify uniqueness of a user, we used the user_id as found in the Stack Overflow database. On average, around four users participated in one thread, and more than one user participated in 2,940 threads (96.4%), and a maximum of 56 distinct users participated in one thread [79].

2. We used sentiment detection algorithm we developed in [78] on the Stack Overflow 2014 data dump

TABLE 2: Statistics of the dataset (A = Answers, C = Comments).

Threads	Posts	A	C	Sentences	Words	Users
3048	22.7K	5.9K	13.8K	87K	1.08M	7.5K
Average	7.46	1.93	4.53	28.55	353.36	3.92

TABLE 3: Distribution of opinionated sentences across APIs.

	Overall			Top Five		
API	**Total**	**+Pos**	**–Neg**	**Total**	**+Pos**	**–Neg**
415	15,627	10,055	5,572	10,330	6,687	3,643
Average	37.66	24.23	13.43	2,066	1,337.40	728.60

From this corpus, we identified the Java APIs that were mentioned in the posts. To identify the Java APIs, we used our API database.

Our API database consists of the Java official APIs and the Java APIs listed in the two software portals Ohloh [80] and Maven central. [81][3] We crawled the Javadocs of five official Java APIs (SE 6-8, and EE 6,7) and collected information about 875 packages and 15,663 types. We consider an official Java package as an API in the absence of any guidelines available to consider otherwise. In total, our API database contains 62,444 distinct Java APIs. All of the APIs (11,576) hosted in Maven central are for Java. From Ohloh, we only included the Java APIs (50,863) out of the total crawled (712,663). We considered a project in Ohloh as a Java API if its main programming language was Java.

We collected the opinionated sentences about APIs using a technique we developed in [50]. The technique works as follows:

1) **Load and preprocess** Stack Overflow posts.
2) **Detect opinionated sentences** using a rule-based algorithm. The algorithm is an adaptation of the Domain Sentiment Orientation (DSO) [82] algorithm for software engineering. Similar adaptation of the algorithm was previously reported by Blair-Goldensohn et al. [83] for Google local product reviews. The algorithm computes sentiment score of each sentence by detecting positive and negative sentiment words in the sentence.
3) **Detection of API names** in the forum texts and hyperlinks based a set of heuristics (e.g., exact and fuzzy matching) and
4) **Association of APIs to opinionated sentences** using heuristics, e.g., proximity between API names and opinionated sentences.

In Table 3, we present summary statistics of the opinionated sentences detected in the dataset. Overall 415 distinct APIs were found. While the average number of opinionated sentences per API was 37.66, it was 2,066 for the top five most reviewed APIs. In fact, the top five APIs contained 66.1% of all the opinionated sentences in the posts. These APIs are jackson, Google Gson, spring framework, jersey, and org.json.

3.3 Reasons for Seeking Opinions About APIs (RQ1)

We aim to understand how and why developers seek and analyze such opinions about APIs and how such information can shape their perception and usage of APIs. The first goal of our study is to learn how developers seek and value the opinions of other developers.

3.3.1 Motivation

By analyzing how and why developers seek opinions about APIs, we can gain insights into the role of API reviews in the daily development activities of software developers. The first step towards

3. We crawled Maven in March 2014 and Ohloh in December 2013.

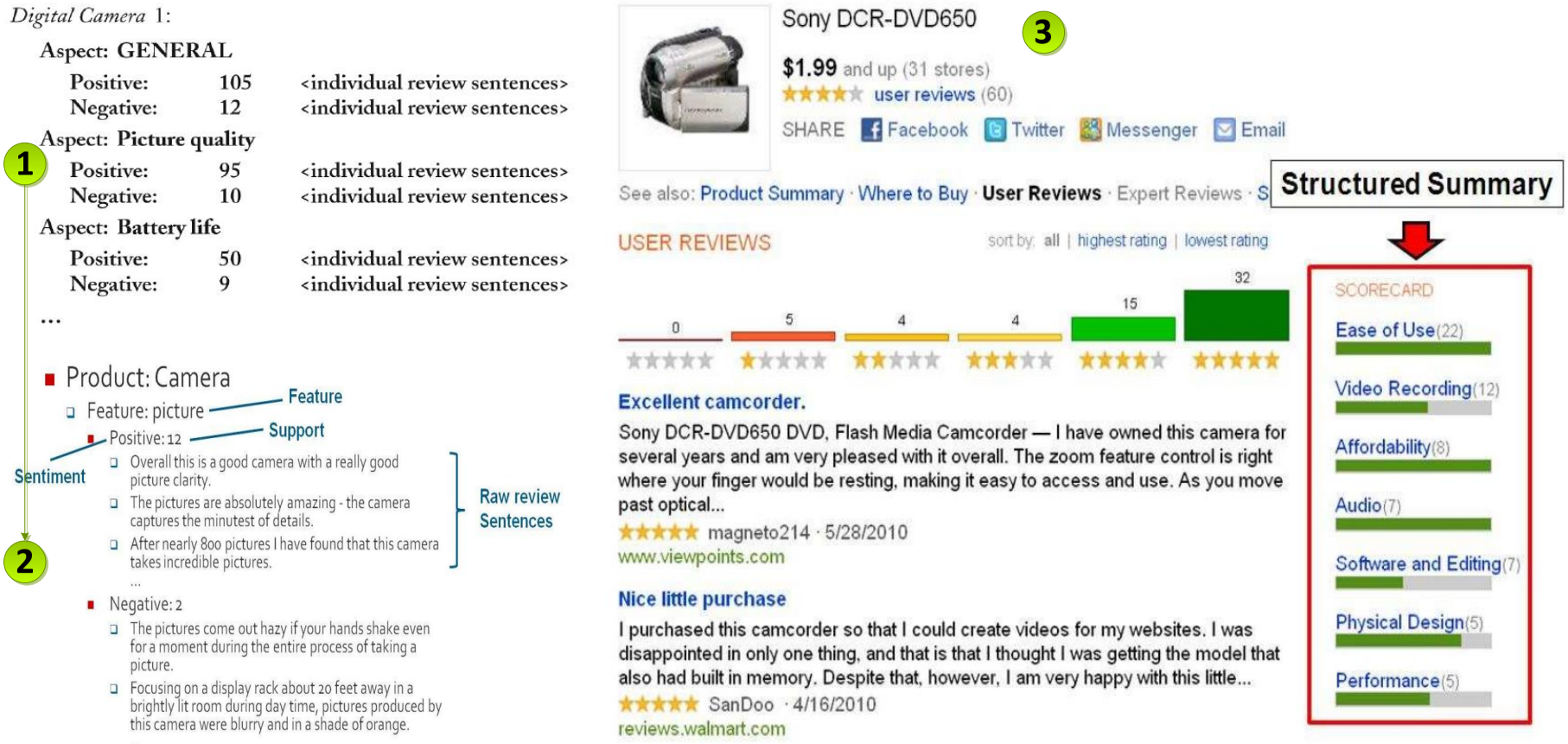

Fig. 3: Screenshots of opinion summarization engine for camera reviews. The different aspects (e.g., picture quality) are used to present summarized viewpoints about the camera. The circle 3 shows an incarnation of the camera product reviews in the now defunct Bing product search. The screenshots are taken from [47].

TABLE 4: The formulation of the research questions in the study.

RQ1	**How do developers seek and value opinions about APIs?**	
1.1	***Theme - Opinion Needs: How do developers seek opinions about APIs?***	
	1.1.a	*Where do developers seek opinions?*
	1.1.b	*What motivates developers to seek opinions about APIs?*
	1.1.c	*What challenges do developers face while seeking API reviews?*
1.2	***Theme - Opinion Quality: How do developers assess the quality of the provided opinions about APIs?***	
RQ2	**How can the support for automated processing of opinions assist developers to analyze API reviews?**	
2.1	***Theme - Tool Support: What tool support is needed to help developers to assess API reviews?***	
2.2	***Theme - Summarization Needs: What are the developers' needs for summarization of API reviews?***	
	2.2.a	*What problems in API reviews motivate the need for summarization?*
	2.2.b	*How summarization of API reviews can support developer decision making?*
	2.2.c	*How do developers expect API reviews to be summarized?*

understanding the developers needs, is to learn about the resources they currently leverage to seek information and opinions about APIs. The developers may use diverse resources to seek opinions about APIs. If a certain resource is used more than other resources, the analysis of the resource can be given more priority over others.

As we noted in Section 3.2, our observation of the API reviews shows that opinions about APIs are shared in the developer forums. Therefore, an understanding of how the different development activities can be influenced and supported through the API reviews can provide us with insights about the factors that motivate developers to seek opinions as well as the challenges that they may face during this process. By learning about these challenges while seeking reviews about APIs, we can gain insights into the complexity of the problem that needs to be addressed to assist developers in their exploration of API reviews.

Finally, opinions are by themselves *subjective*, i.e., opinions about APIs stem from the personal belief or experience of the developers who use the APIs. Therefore, developers may face challenges while assessing the validity of claims by other developers. By analyzing what factors can hinder and support the developers' assessment of the quality of the provided API reviews, we can gain insights into the challenges developers face while leveraging the API reviews.

3.3.2 Approach

We examine developer needs for API reviews via the following questions:

- RQ1.1: How do developers seek opinions about APIs?
- RQ1.2: How do developers assess the quality of the provided opinion?

To exhaustively answer these questions, we further divide RQ1.1 into three sub-questions:

- *RQ1.1.a: Where do developers seek opinions about APIs?*
- *RQ1.1.b: What motivates developers to seek opinions?*
- *RQ1.1.c: What challenges do developers face while seeking for opinions?*

3.4 Tool Support to Analyze API Reviews (RQ2)

The second goal of our study is to understand the needs for tool support to facilitate automated analysis of API reviews.

3.4.1 Motivation

To understand whether developer needs any tools to analyze API reviews, we first need to understand what tools developers may be using currently to analyze the reviews and what problems they may be facing. Such analysis can offer insights into how research in this direction can offer benefits to the developers through future prototypes and tool supports. A predominant direction in the automated processing of reviews in other domains (e.g., cars, cameras, products) is to summarize the reviews. For the domain of API reviews, it can also help to know how developers determine the needs for opinion summarization about APIs. The first step is to determine the feasibility of the existing cross-domain opinion summarization techniques to the domain of API reviews. Such analysis can provide insights into how the summarization approaches adopted in other domains can be applicable to the domain of API reviews. By learning about the specific development needs that can be better supported though the summarization of API reviews, we can gain insights into the potential use cases API review summaries can support. By understanding how developers expect to see summaries of API reviews, we can gain insights into whether and how different summarization techniques can be designed and developed for the domain of API reviews. It is, thus, necessary to know what specific problems in the API reviews should be summarized and whether priority should be given to one API aspect over another.

3.4.2 Approach

We pose two research questions to understand the needs for tool support to analyze API reviews:

- RQ2.1: What tool support is needed to help developers with analyzing and assessing API-related opinions?
- RQ2.2: What are the developers' needs for summarization of opinion-rich information?

To answer these two questions exhaustively, we further divide RQ2.2 into three sub-questions:

- *RQ2.2.a: What problems in API reviews motivate the needs for summarization?*
- *RQ2.2.b: How can summarization of API reviews support the developer's decision making processes?*
- *RQ2.2.c: How do developers expect API reviews to be summarized?*

3.5 The Surveys

We learn about the developer needs for API reviews and tool support for analyzing the reviews through two surveys. We conducted the first survey as a pilot survey and the second as the primary one. The purpose of the pilot survey is to identify and correct potential

ambiguities in the design of the primary survey. Both the pilot and the primary surveys share the same goals. However, the questions of the primary survey are refined and made more focused based on the findings of the pilot survey. For example, in the pilot survey, we mainly focused on the GitHub developers. We picked GitHub for our pilot survey, because previous research shows that GitHub developers use third-party APIs (e.g., open source APIs) and would like to stay aware of changes in APIs in their development tasks to become remain productive [84]. We focused on open source community because open source development is embraced by both individual developers as well as big and small companies (both as contributors to API development as well API usage). In addition, due to the openness, the community may also be more reliant on developer forums to inform choices. Such interactions can be visible to any other developers for further analysis (e.g., during their decision making about an API). This offers us access to diverse viewpoints (i.e., opinions) of developers on APIs that may be used in diverse development needs and contexts.

We found that most of the respondents in our pilot survey considered the developer forums (e.g., Stack Overflow) as the primary source of opinions about APIs. Therefore, in the primary survey, our focus was to understand how developers seek and analyze opinions in developer forums. Stack Overflow is arguably the most popular online forums to share and discuss code and opinions about open-source APIs. Therefore, in our primary survey, we picked developers who are actively involved in the discussions of Stack Overflow posts.

In Section 4, we discuss the design and the summary of results of the pilot survey. In Sections 5 and 6, we discuss the design and detailed results of the primary survey.

4 The Pilot Survey

The pilot survey consisted of 24 questions: three demographic questions, eight multiple-choice, five Likert-scale questions, and eight open-ended questions. In Table 5, we show all the questions of the pilot survey (except the demographic questions) in the order they appeared in the survey questionnaires. The demographic questions concern the participants role (e.g., software developer or engineer, project manager or lead, QA or testing engineer, other), whether they are actively involved in software development or not, and their experience in software development (e.g., less than 1 year, more than 10 years, etc.). The survey was hosted in Google forms and can be viewed at https://goo.gl/forms/8X5jKDKilkfWZT372.

4.1 Pilot Survey Participants

We sent the pilot survey invitations to 2,500 GitHub users. The 2,500 users were randomly sampled from 4,500 users from GitHub. The 4,500 users were collected using the GitHub API. The GitHub API returns GitHub users starting with an ID of 1. We stopped calling the API after it returned the first 4,500 users. From the number of emails sent to GitHub users, 70 emails were bounced back for various reasons, e.g., invalid (domain expired) or non-existent email addresses, making it 2,430 emails being actually delivered. A few users emailed us saying that they were not interested in participating due to the lack of any incentives. Finally, a total of 55 developers responded. In addition, we sent the invitation to 11 developers in a software development company in Ottawa, Canada. The company was selected based on a personal contact we had within the company. The company was involved in multiple different software projects involving open-source APIs. Out of the 11, nine responded. Among the GitHub participants:

1) 78% of respondents said that they are software developers (11% are project managers and 11% belong to "other" category),
2) 92% are actively involved in software development, and
3) 64% have more than 10 years of software development experience, 13% of them have between 7 and 10 years of experience, 9% between 3 to 6 years, 8% between 1 to 2 years and around 6% less than 1 year of experience.

Among the nine Industrial participants, three were team leads and six were professional developers. All nine participants were actively involved in software development. The nine participants had professional development experience between five to more than 10 years.

4.2 Pilot Survey Data Analysis

We analyzed the survey data using statistical and qualitative approaches. For the open-ended questions, we applied an open coding approach [85]. Open coding includes labelling of concepts/categories in textual contents based on the properties and dimensions of the entities (e.g., an API) about which the contents are provided. In our open coding, we followed the card sorting approach [86]. In card sorting, the textual contents are divided into *cards*, where each card denotes a *conceptually coherent quote*. For example, consider the following sentence in Figure 1 (from answer circled as ⑤): *"Note that Jackson fixes these issues, and is faster than GSON."* The sentence has two different conceptual coherent quotes, "Note that Jackson fixes these issues", and "and is faster than GSON". The first quote refers to fixes to issues (i.e., bugs) by the API. The second quote refers to the "performance" aspect of the API. In our analysis, as we analyzed the quotes, themes and categories emerged and evolved during the open coding process.

We created all of the "cards", splitting the responses for eight open-ended questions. This resulted into 173 individual quotes; each generally corresponded to individual cohesive statements. In further analysis, the first two authors acted as coders to group cards into themes, merging those into categories. We analyzed the responses to each open-ended question in three steps:

1) The two coders independently performed card sorts on the 20% of the cards extracted from the survey responses to identify initial card groups. The coders then met to compare and discuss their identified groups.
2) The two coders performed another independent round, sorting another 20% of the quotes into the groups that were agreed-upon in the previous step. We then calculated and report the coder reliability to ensure the integrity of the card sort. We selected two popular reliability coefficients for nominal data: percent agreement and Cohen's Kappa [87].
   Coder reliability is a measure of agreement among multiple coders for how they apply codes to text data. To calculate agreement, we counted the number of cards for each emerged group for both coders and used ReCal2 [88] for calculations. The coders achieved the *almost perfect* degree of agreement; on average two coders agreed on the coding of the content in 96% of the time (the average percent agreement varies across the questions and is within the range of 92–100%; while the average Cohen's Kappa score is 0.84 ranging between 0.63–1 across the questions).

TABLE 5: Pilot survey questions with key highlights from responses. Subscript with a question number shows number of responses.

RQ1.1	**Opinion Needs**
1_{60}	Do you value opinion of other developer when deciding on what API to use? *(yes / no)* ***1) Yes 95%, 2) No 5%***
2_{60}	Where do you seek help/opinions about APIs? *(five sources and others)* ***1) Developer Forums 83.3%, 2) Co-worker 83.3%, 3) Mailing List 33.3%, 4) IRC 26.7%, 5) Others 35%***
3_{60}	How often do you refer to online forums (e.g, Stack Overflow) to get information about APIs? *(five options)* ***1) Every day 23.3%, 2) Two/three times a week 36.7%, 3) Once a month 25%, 4) Once a week 10%, 5) Never 5%***
4_{60}	When do you seek opinions about APIs? *(5-point Likert scale for each option)* ***1) Select among choices 83.3%, 2) Select API version 18.3%, 3) Improve a feature 61.7%, 4) Fix bug 63.3%, 5) Determine a replacement 73.3%, 6) Validate a selection 45%, 7) Develop a competing API 58.3%, 8) Replace an API feature 51.7%***
5_{21}	What are other reasons of you referring to the opinions about APIs from other developers? *(text box)* ***1) Opinion trustworthiness analysis 18.8% 2) API usage 15.6%, 3) API suitability 15.6%, 4) API maturity 6.3%, 5) Usability analysis 6.3%***
6_{28}	What is your biggest challenge while seeking opinions about an API? *(text box)* ***1) Trustworthiness analysis 19.4% 2) Too much info 16.1%, 3) Biased opinion 12.9%, 4) Documentation 9.7%, 5) Staying aware 6.5%***
RQ1.2	**Opinion Quality**
20_{60}	How do you determine the quality of a provided opinion in a forum post? (e.g., Stack Overflow) *(seven options)* ***1) Date 55%, 2) Votes 66.7%, 3) Supporting links 70%, 4) User profile 45%, 5) Code presence 73.3%, 6) Post length 16.7%, 7) Others 1.7%***
21_{15}	What other factors in a forum post can help you determine the quality of a provided opinion? *(text box)* ***1) Documentation 63.3%, 2) Expertise 10.5%, 3) Trustworthiness 10.5%, 4) Situational relevance 10.5%, 5) Biased opinion 5.3%***
RQ2.1	**Tool Support**
7_{60}	What tools can better support your understanding of opinions about APIs in online forum discussions? *(5-point Likert scale for each option)* ***1) Opinion mine & summarize 68.3%, 2) Sentiment mine 45%, 3) Comparator analyze 73.3%, 4) Trends 56.7%, 5) Co-mentioned APIs 76.7%***
8_{17}	What other tools can help your understanding of opinions about APIs in online forum discussions? *(text box)* ***1) Reasoning 18.2%, 2) Extractive summaries 18.2%, 3) Dependency info 12.1%, 4) Documentation 9.1%, 5) Expertise find 9.1%***
RQ2.2	**Summarization Needs**
9_{60}	How often do you feel overwhelmed due to the abundance of opinions about an API? *(four options)* ***1) Every time 5%, 2) Some time 48.3%, 3) Rarely 33.3%, 4) Never 13.3%***
10_{60}	Would summarization of opinions about APIs help you to make a better decision on which one to use? *(yes/no)* ***1) Yes 83.3%, 2) No 16.7%***
11_{59}	Opinions about APIs need to be summarized because? *(5-point Likert scale for each option)* ***1) Too many posts with opinions 60%, 2) Interesting opinion in another post 66.7%, 3) Contrastive viewpoints missed 55%, 4) Not enough time to look for all opinions 56.7%***
12_{60}	Opinion summarization can improve the following decision making processes. *(5-point Likert scale for each option)* ***1) Select among choices 85%, 2) Select API version 45%, 3) Improve a feature 48.3%, 4) Fix bug 40%, 5) Determine a replacement 78.3%, 6) Validate a selection 53.3%, 7) Develop a competing API 48.3%, 8) Replace an API feature 46.7%***
13_{9}	What other areas can be positively affected having support for opinion summarization? *(text box)* ***1) API usage 21.4%, 2) Trustworthiness analysis 21.4%, 3) Documentation 9.5%, 4) Maturity analysis 14.3%, 5) Expertise 7.1%***
14_{9}	What other areas can be negatively affected having support for opinion summarization? *(text box)* ***1) Trustworthiness analysis 33.3%, 2) Info overload 33.3%, 3) API usage 8.3%, 4) Reasoning 8.3%***
15_{60}	An opinion is important if it contains discussion about one or more of the following API aspects? *(5-point Likert scale for each option)* ***1) Performance 86.7%, 2) Security 85%, 3) Usability 85%, 4) Documentation 88.3%, 5) Compatibility 75%, 6) Community 73.3%, 7) Bug 86.7%, 8) Legal 43.3%, 9) Portability 50%, 10) General Features 48.3%, 11) Only sentiment 15%***
16_{16}	What are the other factors you look for in API opinions? *(text box)* ***1) API usage 28.6%, 2) Usability 19%, 3) Expertise 14.3%, 4) Standards 9.5%, 5) Reasoning 9.5%***
17_{60}	What type of opinion summarization would you find most useful? *(five types)* ***1) In a paragraph 83.3%, 2) Divided into aspects 78.3%, 3) Top N by most recent 36.7%, 4) Divided into topics 35%, 5) Others 1.7%***
18_{60}	How many keywords do you find to be sufficient for topic description? *(five options)* ***1) Five words 61.7%, 2) 5-10 words 28.3%, 3) 10-15 words 5%, 4) 15-20 words 3.3%, 5) Not helpful at all 1.7%***
19_{7}	What are the other ways opinions about APIs should be summarized? *(text box)* ***1) API Usage 25%, 2) Expertise 8.3%, 3) Documentation 8.3%, 4) Popularity 8.3%, 5) Testing 8.3%***

3) The rest of the card sort (for each open-ended question), i.e., 60% of the quotes, was performed by both coders together.

4.3 Summary of Results from the Pilot Survey

In this section, we briefly discuss the major findings from the pilot survey, that influenced the design of the primary survey. A detail report of the findings is in our online appendix [89]. Out of the 64 respondents who completed the demographic questions, maximum 60 respondents answered the other questions.

In Table 5, the type of each question (e.g., open or closed-ended) is indicated beside the question (in *italic* format). The *key findings* for each question are provided under the question in Table 5 (in ***bold italic*** format). The key findings are determined as follows:

Responses to open-ended questions. The response to an open-ended question is provided in a text box (e.g., Q5). For each such question, we show the top five categories (i.e., themes) that emerged from the responses of the question. The frequency of categories under a question is used to find

the top five categories for the question.

Likert-scale questions Each such question can have multiple options for a user to choose from (e.g., Q7). Each option can be rated across five scales: Strongly Agree, Agree, Neutral, Disagree, and Strongly Disagree. For each question, we calculate the percentage of respondents that agreed (= Total Agreed + Total Strongly Agreed) with the provided option.

Multiple choice questions. There were two types of multiple choice questions. 1) Only Select One: Each such question can have more than one choice and a participant can pick one of the the options (e.g., Q1). 2) Can Select Multiple. The participant can pick one, more than one, or all the choices, when the options are of type multiple select (e.g., Q17). For each option, we calculate the percentage of respondents that picked the option.

Needs for API Reviews (RQ1.1). 95% of the participants reported that they consider opinions of other developers while making a selection of an API. The primary sources of such opinions for them are both developer forums and co-workers (83.3%). 69.3% of the participants reported to have consulted developers forums to gain information about APIs. Most of the participants (83.3%) reported that they consider opinions about APIs while making a selection of an API among multiple choices. They also seek opinions to determine a replacement of an API (73.3%), as well as fixing a bug (63.3%), etc. Among the other reasons for the participants to seek opinion were the learning of API usage and the analysis of validity of a provided API usage (e.g, if it actually works). The participants reported that making an informed decision among too much opinion, and judging the trustworthiness of the provided opinions are the major challenges they face while seeking and analyzing opinions.

Needs for API Review Quality Analysis (RQ1.2). Most of the participants (73.3%) considered an opinion, accompanied by a code example, to be of high quality (Q20). 70% of the participants considered the presence of links to supporting documents as a good indicator of the quality of the provided opinion, while 55% believed that the posting date is an important factor. Stack Overflow uses upvotes and downvotes of a post as a gauge of the public opinion on the content. 66.7% of developers attributed a higher count of votes on the post to their assumption of the higher quality of the corresponding opinion in the post. The user profile, another Stack Overflow feature, was found as useful by 45% of participants to evaluate the quality of the provided opinion. The length of the post where the opinion is provided was not considered as a contributing factor during the assessment of the opinion quality (agreed by only 16.7% of the responders). One participant did not agree with any of the choices, while two participants mentioned two additional factors: (1) real example, i.e., the provided code example should correspond to a real world scenario; and (2) reasoning of the provided opinion.

Tools for API Review Analysis (RQ2.1). More than 83% of the respondents agreed that opinion summarization is much needed for several reasons. Since opinions change over time or across different posts, developers wanted to be able to track changes in API opinions and to follow how APIs evolve. In our surveys, we sought to explain each option in the closed questions with a short description or examples. The description for each question can be found in the links to the surveys (see https://goo.gl/forms/8X5jKDKilkfWZT372 for the pilot survey). For example, in Q11 of the pilot survey (Table 5), the two options "Opinions can change over time" and "Opinions can evolve over time" are differentiated as follows: Opinions about an API about a feature can change from bad to good, e.g., developers did not like it before, but like it now (i.e., we need a contrastive viewpoint). In contrast, *overall* opinions about an API can evolve, e.g., it is more positive now due to the increase in adoption by users (e.g., by becoming more usable), etc.

Tools for API Review Summarization (RQ2.2). The vast majority of responders also thought that an interesting opinion about an API might be expressed in a post that they have missed or have not looked at. The need for opinion summarization was also motivated by the myriad of opinions expressed via various developer forums. Developers believe that the lack of time to search for all possible opinions and the possibility of missing an important opinion are strong incentives for having opinions organized in a better way.

While developers were interested in tool support for mining and summarizing opinions about APIs, they also wanted to link such opinions to other related dimensions, such as, usage, maturity, documentation, and user expertise. 85% of the respondents believed that opinion summarization could help them select an API among multiple choices. More than 78% believed that opinion summaries can also help them find a replacement to an API. Developers agreed that summaries can also help them develop a new API to address the needs that are currently not supported, improve a software feature, replace an API feature, validate the choice of an API, select the right version of an API, and fix a bug (48.3%, 48.3%, 46.7%, 53.3%, 45%, and 40%, respectively).

4.4 Needs for the Primary Survey

The responses in our pilot survey showed that opinions about APIs are important in diverse development needs. However, the survey had the following limitations:

Design.

A number of similar questions were asked in pairs. For example, in Q4, we asked the developers about the reasons to seek opinions. The developers were provided eight options to choose from. In Q5, we asked the developers to write about the other reasons that could also motivate them to seek opinions. Q5 is an open-ended question. Both Q4 and Q5 explore similar themes (i.e., needs for opinion seeking). Therefore, the responses in Q5 could potentially be biased due to the respondents already being presented the eight possible options in Q4. A review of the manuscript based on the pilot survey (available in our online appendix [89]) both by the colleagues and the reviewers in Transaction of Software Engineering pointed out that a better approach would have been to ask Q5 before Q4, or only ask Q5. Moreover, the respondent should not be given an option to modify his response to an open-ended question, if a similar themed closed-ended question is asked afterwards.

Sampling.

We sampled 2,500 GitHub developers out of the first 4,500 GitHub IDs as returned by the GitHub API. We did not investigate the relevant information of the developers, e.g., are they still active in software development? Do they show expertise in a particular programming languages?, and so on. This lack of background information on the survey population can also prevent us from making a formal conclusion out of the responses of the survey.

Response Rate.

While previous studies involving GitHub developers also reported low response rate (e.g., 7.8% by Treude et al. [84]), the response rate in our pilot survey was still considerably lower (only 2.62%). There can be many reasons for such low response rate. For example, unlike Treude et al. [84], we did not offer any award/incentives to participate in the survey. However, our lack of enough knowledge of the GitHub survey population prevents us from making a definitive connection between the low response rate and the lack of incentives.

We designed the primary survey to address the above limitations. Specifically, we took the following steps to avoid the above problems in our primary survey:

1) We asked all the open-ended questions before the corresponding closed-ended questions. The respondents only saw the closed-ended questions after they completed their responses to all the open-ended questions. The respondents were not allowed to change their response to any open-ended question, once they were asked the closed-ended questions.
2) We conducted the primary survey with a different group of software developers, all collected from Stack Overflow. To pick the survey participants, we applied a systematic and exhaustive sampling process (discussed in the next section).
3) We achieved a much higher response rate (15.8%) in our primary survey.

In the next section, we discuss in details about the design of the primary survey. In Section 7, we briefly compare the results of the pilot and primary survey on the similar-themed question pairs.

5 Primary Survey Design

We conducted the primary survey with a different group of software developers. Besides the three demographic questions, the primary survey contained 24 questions. In Table 6, we show the questions in the order they were asked. We show how the questions from the pilot survey were asked in the primary survey in the last column of Table 6. The survey was conducted using Google forms and is available for view at: https://goo.gl/forms/2nPVUgBoqCcAabwj1.

As we noted in Section 3.5, in our primary survey, we focused on understanding how and whether developers seek and analyze API reviews in developer forums. This decision was based on the observations from our pilot survey. The participants in our pilot survey reported the developer forums as their primary sources for seeking opinions about APIs (along with co-workers). One of our major goals from the surveys is to elicit requirements for tool designs to facilitate API analysis using API reviews. Developer forums, such as Stack Overflow, can be a sharing place for co-workers as well. Moreover, the design and deployment of such tools can be better facilitated if the data is already available and shared in the forum posts.

In Table 6, the horizontal lines between questions denote sections. For example, there is only one question in the first section (Q1). Once a developers responds to all the questions in a section, he is navigated to the next section. Depending on the type of the question, the next section is determined. For example, if the answer to first question ("Do you visit developer forums to seek info about APIs?") is a 'No', we did not ask him any further questions. If the answer is a 'Yes', the respondent is navigated to the second question. The navigation between the sections was designed to ensure two things: 1) that we do not ask a respondent irrelevant questions. For example, if a developer does not value the opinions of other developers, it is probably of no use asking him about his motivation for seeking opinions about APIs anyway, and 2) that the response to a question is not biased by another relevant question. For example, the first question in the third section of Table 6 is Q4 ("What are your reasons for referring to opinions of other developers about APIs in online developer forums?"). This was an open-ended question, where the developers were asked to write their responses in a text box. A relevant question was Q18 in the sixth section ("When do you seek opinions about APIs?"). The developers were given eight options in a Likert scale (e.g., selection of an API among choices). The developers were able to answer Q18 only after they answered Q4. The developers were not allowed to return to Q4 from Q18. We adopted similar strategy for all such question pairs in the primary survey. In this way, we avoided the problem of potential bias in the developers' responses in our primary survey.

The second last column in Table 6 shows how the questions are mapped to the two research questions (and the sub-questions) that we intend to answer. The pilot and the primary surveys contained similar questions. The last column of Table 6 shows how 18 of the 24 questions in the primary survey were similar to 18 questions in the Pilot survey. While the two sets of questions are similar, in the primary survey the questions focused specifically on developer forums. For example, Q4 in the primary survey (Table 6) was "What are your reasons for referring to opinions of other developers about APIs in online developer forums?" The similar question in the pilot survey was Q5 (Table 5): "What are other reasons for you to refer to the opinions about APIs from other developers?"

To ensure that we capture developers' experience about API reviews properly in the primary survey, we also asked six questions that were not part of the pilot survey. The first two questions (discussed below) in the primary survey were asked to ensure that we get responses from developers who indeed seek and value opinions about APIs. The first question was "Do you visit developer forums to seek info about APIs?". If a developer responded with a 'No' to this question, we did not ask him any further questions. We did this because 1) developer forums are considered as the primary resource in our pilot survey and 2) automated review analysis techniques can be developed to leverage the developer forums. The second new question was about asking the participants about the top developer forums they recently visited. Such information can be useful to know which developer forums can be leveraged for such analysis. The third new question was "Do you value the opinion of other developers in the forum posts while deciding on what API to use?". If the response was a 'no', we asked the participant only one question (24): "Please explain why you don't value the opinion of developers in the forum posts". We asked this question to understand the potential problems in the opinions that may be preventing them from leveraging those opinions. In Figure 4, we show how the above two questions are used to either navigate into the rest of the survey questions or to complete the survey without asking the respondents further questions.

5.1 Participants

We targeted developers that participated in the Stack Overflow forum posts (e.g., asked a question or provided an answer to a question in Stack Overflow). Out of a total of 720 invitations sent,

TABLE 6: Questions asked in the primary survey. The horizontal lines between questions denote sections

No	Question	Theme	Map (Pilot)
1	Do you visit developer forums to seek info about APIs? *(yes/no)*	RQ1.1.a	
2	List top developer forums you visited in the last two years *(text box)*	RQ1.1.a	
3	Do you value the opinion of other developers in the developer forums when deciding on what API to use? *(yes/no)*	RQ1.2	1
4	What are your reasons for referring to opinions of other developers about APIs in online developer forums? *(text box)*	RQ1.1.b	5
5	How do you seek information about APIs in a developer forum? How do you navigate the multiple posts?	RQ1.1.c	
6	What are your biggest challenges when seeking for opinions about an API in an online developer forum? *(text box)*	RQ1.1.c	6
7	What factors in a forum post can help you determine the quality of a provided opinion about an API? *(text box)*	RQ1.2	21
8	Do you rely on tools to help you understand opinions about APIs in online forum discussions? *(yes/no)*	RQ2.1	
9	If you don't use a tool currently to explore the diverse opinions about APIs in developer forums, do you believe there is a need of such a tool to help you find the right viewpoints about an API quickly? *(yes/no)*	RQ2.1	
10	You said yes to the previous question on using a tool to navigate forum posts. Please provide the name of the tool. *(text box)*	RQ2.1	8
11	What are the important factors of an API that play a role in your decision to choose an API? *(text box)*	RQ2.2.c	16
12	Considering that opinions and diverse viewpoints about an API can be scattered in different posts and threads of a developer forum, what are the different ways opinions about APIs can be summarized from developer forums? *(text box)*	RQ2.2.c	19
13	What areas can be positively affected by the summarization of reviews about APIs from developer forums? *(text box)*	RQ2.2.b	13
14	What areas can be negatively affected by the summarization of reviews about APIs from developer forums? *(text box)*	RQ2.2.b	14
15	An opinion is important if it contains discussion about the following API aspects? *(5-point Likert scale for each option)*	RQ2.2.c	15
16	Where do you seek help/opinions about APIs? *(5 opinions + None + Text box to write other sources)*	RQ1.1.a	2
17	How often do you refer to online forums (e.g., Stack Overflow) to get information about APIs? *(five options)*	RQ1.1.a	3
18	When do you seek opinions about APIs? *(5-point Likert scale for each option)*	RQ1.1.b	4
19	What tools can better support your understanding of API reviews in developer forums? *(5-point Likert scale for each option)*	RQ2.1	7
20	How often do you feel overwhelmed due to the abundance of opinions about an API? *(four options)*	RQ2.2.a	9
21	Would a summarization of opinions about APIs help you to make a better decision on which one to use? *(yes/no)*	RQ2.2.a	10
22	Opinions about an API need to be summarized because *(5-point Likert scale for each option)*	RQ2.2.a	11
23	Opinion summarization can improve the following decision making processes *(5-point Likert scale for each option)*	RQ2.2.b	12
24	Please explain why you don't value the opinion of other developers in the developer forums. *(text box)*	RQ1.2	

TABLE 7: Summary statistics of the primary survey population and sampled users from the population.

	Total	Threads	Created			Last Active			Reputation				
	Users	Contributed	2012	2011	<= 2010	2017	2016	<= 2015	Min	Max	Median	Avg	Std
Population	88,021	3,233,131	16,080	26,257	45,684	40,780	10,409	36,832	1	627,850	159	1,866.5	10,169.1
Sample	900	772,760	47	151	702	900	0	0	5,471	627,850	18,433.5	43,311.22	66,738

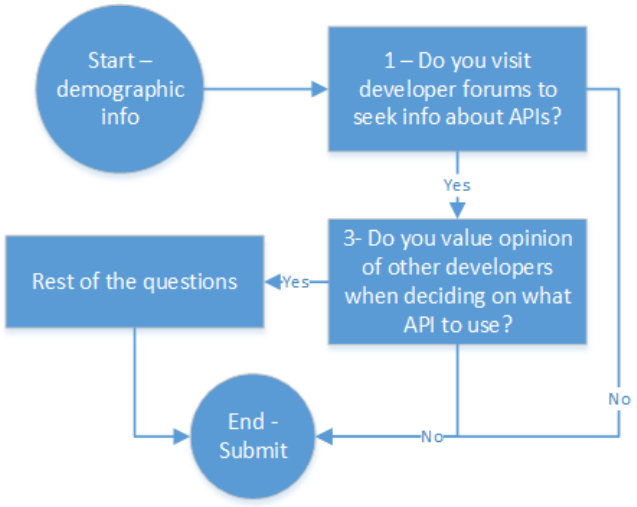

Fig. 4: Navigation between sections in the primary survey.

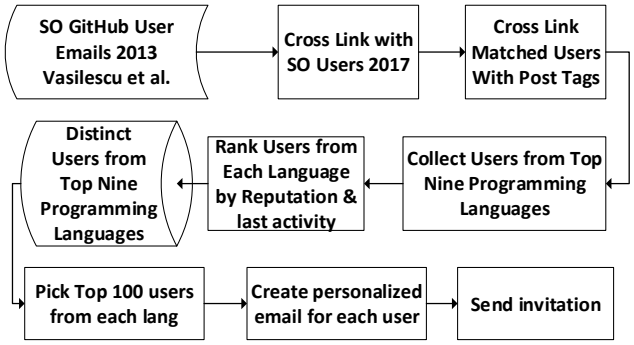

Fig. 5: Sampling of participants for the primary survey.

we received 114 responses (response rate 15.8%). Among those, 72.8% responded that they visit developers forums and they value the opinion of other developers in the forums. The distribution of the profession of those participants is: 1) Software developers 79.5%, 2) Tech/Team Lead 13.3%, 3) Research Engineer 3.6%, 4) Student 3.6%, and 5) Other 1.2% . The distribution of experience of the participants is: 1) 10+ years 56.6%, 2) Seven to 10 years 28.9%, and 3) Three to six years 14.5%. To report the experience, the developers were given five options to choose from (following Treude et al. [84]): 1) less than 1 year, 2) 1 to 2 years, 3) 3 to 6 years, 4) 7 to 10 years, and 5) 10+ years. None of the respondents reported to have software development experience of less than three years. Therefore, we received responses from experienced developers in our primary survey. 97.6% of them were

actively involved in software development.

5.1.1 Sampling Strategy

To recruit the participants for an empirical study in software engineering, Kitchenham et al. [90] offered two recommendations:

Population *"Identify the population from which the subjects and objects are drawn"*, and

Process *"Define the process by which the subjects and objects were selected"*.

We followed both of the two suggestions to recruit the participants for our primary survey (discussed below).

The contact information of users in Stack Overflow is kept hidden from the public to ensure that the users are not spammed. Vasilescu et al. [91] correlated the user email hash from Stack Overflow to those in GitHub. To ensure that the mining of such personal information is not criticized by the Stack Overflow community in general, Bogdan Vasilescu started a question in Stack Overflow with the theme, which attracted a considerable number of developers from the Stack Overflow community in 2012 [92]. The purpose of Vasilescu et al. [91] was to see how many of the Stack Overflow users are also active in GitHub. In August of 2012, they found 1,295,623 Stack Overflow users. They cross-matched 93,772 of those users in GitHub. For each of those matched users in GitHub, they confirmed their email addresses by mapping their email hash in Stack Overflow to their email addresses as they shared in GitHub. Each user record in their dataset contains three fields: (1) UnifiedId: a unique identifier for each record (an auto-incremental integer), (2) GitHubEmail: the email address of the user as posted in GitHub, (3) SOUserId: the ID of the user in Stack Overflow. We used this list of 93,772 users as the *potential population source* of the primary survey. In our survey invitation, we were careful not to spam the developers. For example, we only sent emails to them twice (the second time as a reminder). In addition, we followed the Canadian Anti-Spam rules [93] while sending the emails, by offering each email recipient the option to 'opt-out' from our invitation. We did not send the second, i.e., reminder email to the users who decided to opt-out.

We sampled 900 users from the 93,772 users as follows (Figure 5):

1) **Match:** Out of the 93,772 users in the list of Vasilescu et al. [91], we found 92,276 of them in the Stack Overflow dataset of March 2017. We cross-linked the user 'ID' in the Stack Overflow dataset to the 'SOUserId' field in the dataset of Vasilescu et al. [91] to find the users. For each user, we collected the following information: (1) Name, (2) Reputation[4], (3) User creation date, and (4) Last accessed date in Stack Overflow.
2) **Discard:** Out of the 92,276 users, we discarded 4,255 users whose email addresses were also found in the list of 4,500 GitHub users that we used to sample the 2,500 GitHub users for the pilot survey. Thus, the size of the target population in our primary survey was 88,021.
3) **Tag:** For each user out of the population (88,021), we then searched for the posts in the Stack Overflow data dump of 2017 where he has contributed by either asking the question or answering the question. For each user, we created a user tag frequency table as follows: a) For each post contributed by the user, we collected the id of the thread of the post. b) For each thread, we collected the list of tags assigned to it. We put all those tags in the tag frequency table of the user. c) We computed the occurrence of each tag in the tag frequency table of the user d) We ranked the tags based on frequency, i.e., the tag with the highest occurrence in the table was put at the top.
4) **Programming Languages:** We assigned each user to one of the following nine programming languages:(1) Javascript, (2) Java, (3) C#, (4) Python, (5) C++, (6) Ruby, (7) Objective-C, (8) C, and (9) R. The nine programming languages are among the top 10 programming languages in Stack Overflow, based on the number of questions tagged as languages. In our survey population, we observed that more than 95% of the users tagged at least one of the languages in the Stack Overflow posts. We assigned a user to a language, if the language had the highest occurrence among the nine languages in the user frequency table of the user. If a user did not have any tag resembling any of the nine languages, we did not include him/her in the sample.
5) **Ranking of Users.** For each language, we ranked the users by reputation and activity date, i.e., first by reputation and then if two users had the same reputation, we put the one at the top between the two, who was active more recently.
6) **Create sample** For each language, we picked the top 100 users. For each user, we created a personalized email and sent him/her the survey invite.

In Table 7, we show the summary statistics of the primary survey population and the sampled users. The 88,021 users contributed to more than 3.2M threads. As of September 2017, Stack Overflow hosts 14.6M threads and 7.8M users. Thus, the users in the population corresponds to 0.012% of all the users in Stack Overflow, but they contributed to 22.1% of all threads in Stack Overflow. All of the 900 users were active in Stack Overflow as early as 2017, even though most of them first created their account in Stack Overflow on or before 2010. Moreover, each of the sampled users was highly regarded in the Stack Overflow community, if we take their reputation in the forum as a metric for that – the minimum reputation was 5,471 and the maximum reputation was 627,850, with a median of 18,434. Therefore, we could expect that the answers from these users would ensure informed insights about the needs for opinions in reviews about APIs posted in Stack Overflow.

5.1.2 Participation Engagement

While targeting the right population is paramount for a survey, convincing the population to respond to the survey is a non-trivial task. As Kitchenham et al. [90] and Smith et al. [94] noted, it is necessary to consider the contextual nature of the daily activities of software developers while inviting them to the survey. While sending the survey invitations, we followed the suggestions of Smith et al. [94] who reported their experience on three surveys conducted at Microsoft. Specifically, we followed five suggestions, two related to *persuasion* (Liking, Authority and credibility) and three related to *social factors* (Social benefit, compensation value, and timing).

Liking. Nisbett and Wilson [95] explained the cognitive bias *halo effect* that people are *more likely to comply with a request from a person they have positive affect afterwards* [94]. Their advice to leverage the positive affection is to communicate with the people in the study by their name. For our survey, we created a personalized email for each user by (i) addressing

4. The reputation of a user in Stack Overflow is based on the votes from other developers

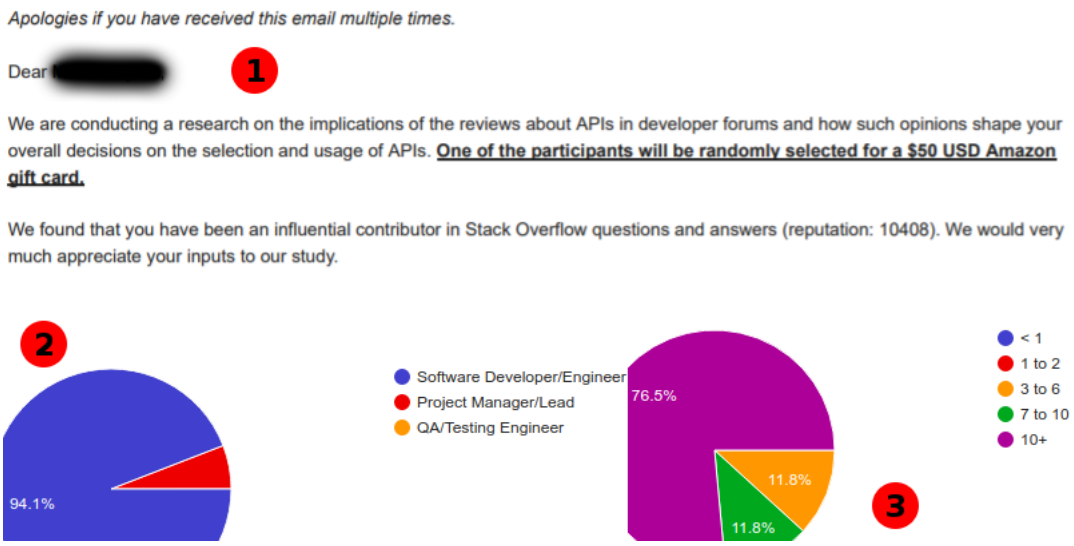

Fig. 6: Snapshots of the primary survey email and responses

the developer by his name in the email, and (ii) including the reputation of the developer in the email. In Figure 6, we show a screenshot of an email sent to one of the users (1).

Authority and credibility. Smith et al. [94] pointed out that the *"compliance rates rise with the authority and credibility of the persuader"*. To ensure that the developers considered our survey as authentic and credible, we highlighted the nature of the research in both the email and the survey front page. We also cited that the survey was governed by the formal Ethics Approval from McGill University and that the reporting to the survey would be anonymized.

Social Benefit. Edwards found that participants are more likely to respond to survey requests from universities [96]. The reason is that, participants may be more likely to respond to a survey if they know that their response would not be used for commercial benefits, rather it will be used to benefit the society (e.g., through research). We contacted the survey participants using the academic emails of all the authors of the paper. We also mentioned in the email that the survey is conducted as a PhD research of the first author, and stressed on the fact that the survey was not used for any commercial purpose.

Compensation value. Smith et al. [94] observed in the surveys conducted at Microsoft that people will likely comply with a survey request if they owe the requester a favor. Such reciprocity can be induced by providing an incentive, such as a gift card. Previous research showed that this technique alone can double the participation rate [97]. In our primary survey, we offered a 50 USD Amazon Gift card to one of the randomly selected participants.

Timing. As Smith et al. [94] experienced, the time when a survey invitation is sent to the developers, can directly impact their likelihood of responding to the email. They advise to avoid sending the survey invitation emails during the following times: 1) Monday mornings (when developers just quickly want to skim through their emails) 2) Most likely out of office days (Mondays, Fridays, December month). We sent the survey invitations only on the three other weekdays (Tuesday-Thursday). We conducted the survey in August 2017.

Out of the 900 emails we sent to the sampled users, around 150 bounced back for various reasons (e.g., old or unreachable email). Around 30 users requested by email that they would not want to participate in the survey. Therefore, the final number of emails sent *successfully* was 720. We note that the email list compiled by Vasilescu et al. [91] is from 2013. Therefore, it may happen that not all of the 720 email recipients may have been using those email addresses any more. Previously, Treude et al. [84] reported a response rate of 7.8% for a survey conducted with the Github developers. Our response rate (15.8%) is more than the response rate of Treude et al. [84].

5.2 Survey Data Analysis

We analyzed the primary survey data using the same statistical and qualitative approaches we applied in the pilot survey. We created a total of 947 quotes from the responses to the open-ended questions. We created quotes from each response as follows: 1) We divided it into sentences. 2) We further divided each sentence into individual clauses. We used semi-colon as a separator between two clauses. Each clause was considered a quote.

Two coders analyzed the quotes. The first author was the first coder. The second coder was selected as a senior software engineer working in the Industry in Ottawa, Canada. The second coder is not an author of this manuscript.

The two coders together coded eight of the nine open-ended questions (Q4-Q7, Q11-Q14 in Table 6). The other open-ended question was Q24 ("explain why you don't value the opinion of other developers ..."). Only 10 participants responded to that question, resulting in 17 quotes. The first coder labelled all of those. In Table 8, we show the agreement level between the two coders for the eight open-ended questions. The last row in Table 8 shows the number of quotes for each of the questions. To compute the agreement between the coders, we used the online recal2 calculator [88]. The calculator reports the agreement using four measures: 1) Percent agreement, 2) Cohen κ [87], 3) Scott's Pi [98], and 4) Krippendorff's α [99] The Scott's pi is extended to more than two coders in Fleiss' κ. Unlike Cohen κ, in Scott's Pi the coders have the same distribution of responses. The Krippendorff's α is more sensitive to bias introduced by a coder, and is recommended over Cohen κ [100]. The agreement (Cohen

TABLE 8: The agreement level between the coders in the open coding.

	Q4	Q5	Q6	Q7	Q11	Q12	Q13	Q14
Percent	98.9	98.6	99.0	96.4	98.9	98.4	100	100
Cohen κ	0.90	0.76	0.80	0.66	0.80	0.75	1	1
Scott π	0.90	0.77	0.80	0.67	0.81	0.76	1	1
Krippen α	0.90	0.77	0.80	0.67	0.81	0.76	1	1
Quotes	112	104	112	121	150	122	101	108

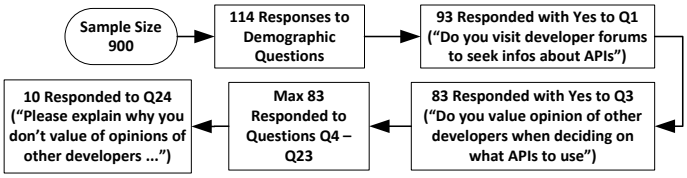

Fig. 7: The number of responses to the questions in the primary survey

κ) between the coders is above 0.75 for all questions except Q7. For Q7, the agreement is above the substantial level [101].

6 Primary Survey Results

During the open-coding process of the nine open-ended questions, 37 categories emerged (excluding 'irrelevant' and responses such as 'Not sure'). We labelled a quote as 'Not Sure' when the respondent in the quote mentioned that he was not sure of the specific answer or did not understand the question. For example, the following quotes were all labelled as 'Not sure': 1) *"Don't know"* 2) *"I am not sure"* 3) *"This is a hard problem."* 4) *"I'm not sure what this question is asking."* The 37 categories were observed a total of 1,019 times in the quotes. 46 quotes have more than one category. For example, the following quote has three categories (Documentation, API usage, and Trustworthiness): *"Many responses are only minimally informed, have made errors in their code samples, or directly contradict first-party documentation."* In Table 19 (Appendix A), we show the distribution of the categories by the questions. In Figure 7, we show the number of participants that answered the different questions in the primary survey. We discuss the results below.

6.1 Reasons for Seeking Opinions about APIs (RQ1.1)

We report the responses of the developers along the three sub-questions: 1) Sources for development and opinion needs, 2) Factors motivating developers to seek opinion about APIs from developer forums, and 3) Challenges developers face while seeking for opinions about APIs from developer forums.

6.1.1 Sources for Opinions about APIs (RQ1.1.a)

We asked the developers three questions (Q2, 16, 17 in Table 6).

Q2 *We asked them to give a list of the top developer forums they visited in the last two years*. The respondents reported 40 different developer forums. Stack Overflow and its companion sites in Stack Exchange were reported the most (94 times), at least once by each respondent. The other major forums listed were (in order of frequency): 1) GitHub issue tracker (9), 2) Google developer forum (8), 3) Apple developer forum (6), 4) Twitter/Quora/Reddit/Slack (3) The other mentions were blog lists, internal mailing lists, XDA, Android forums, etc.

Q16 *We further asked them which sources they use to seek opinions about APIs*. We gave them six options to choose from: 1) Developer forums, e.g., Stack Overflow (Picked by all 83 developers who mentioned that they valued the opinions of other developers and that they visit developer forums). 2) Co-workers (63 developers), 3) IRC chats (22 developers), 4) Internal mailing lists (24 developers), 5) None (0 developers), 6) Others. For 'Others', we gave them a text box to write the name of the forums. Among the other sources, developers picked a variety of online resources, such as, Google search engine, Hacker news, blogs, Slack (for meetups), GitHub, Twitter, etc.

The Google and Apple developer forums were present in the list of forums that the developers visited in the last two years, but they were absent in the list of forums where developers visit to seek opinions. Stack Overflow was picked in both questions as the most visited site by the developers both as a general purpose forum to find information about APIs and as a forum to seek opinions about APIs. We observed the presence of Twitter in both lists (Q2 and Q16). Therefore, besides Stack Overflow, Twitter can be another resource to support developer's learning needs, as was previously observed by Sharma et al. [102].

Q17 *We asked the developers about their frequency of visiting the online developer forums (e.g., Stack Overflow) to get information about APIs (Q17)*. There were six options to choose from: 1) Every day (picked by 36.1% of the developers), 2) Two/three times a week (32.5%), 3) Once a week (14.5%), 4) Once a month (16.9%), 5) Once a year (0%), 6) Never (0%). Therefore, most of the developers (83.1%) reported that they visit developer forums at least once a week, with the majority (36.1%) visiting the forums every day. Each developer reported to visit the developer forums to seek opinions about APIs at least once a month.

Reasons for Seeking Opinions about APIs (RQ1.1)
Sources for Opinions about APIs (RQ1.1.a)

Stack Overflow was considered as the major source to seek information and opinions about APIs. Developers also use diverse informal documentation resources, e.g., blogs, Twitter, GitHub issue tracking system, etc.

6.1.2 Factors Motivating Opinion Seeking (RQ1.1.b)

We asked the developers two questions (Q4,18 in Table 6). For each category as we report below (e.g., Expertise^{31,31} below), the superscript (n, m) is interpreted as follows: n for the number of quotes found in the responses of the questions, and m is the number of total distinct participants provided those responses. A similar format was previously used by Treude et al. [84] (except m, i.e., number of respondents).

Q4 *We asked the developers to write about the reasons for them to refer to opinions about APIs from other developers*. The developers seek opinions for a variety of reasons:

1) To gain overall Expertise^{31,31} about an API by learning from experience of others. Developers consider the opinions from other expert developers as indications of real-word experience and hope that such experience can lead them to the right direction, *"Getting experience of others can save a lot of time if you end up using a better API or end up skipping a bad one."*.

2) To learn about the USAGE[17,16] of an API by analyzing the opinion of other developers posted in response to the API usage scenarios in the posts. Developers consider that certain edge cases of an API usage can only be learned by looking at the opinions of other developers, *"It's especially helpful to learn about problems others' faced that might only be evident after consider time is spent experimenting with or using the API in production."* They expect to learn about the potential issues about an API feature from the opinions, before they start using an API, *"Because there is always a trick and even best practice which can only be provided by other developers."*
3) To be able to SELECT[13,12] an API among multiple choices for a given development. The developers think that the quality of APIs can vary *considerably*. Moreover, not all APIs may suit the needs at hand. Therefore, they expect to see the pros and cons of the APIs before making a selection. *"If I don't know an API and I need to pick one, getting the opinion of a few others helps make the decision."* To make a selection or to meet the specific development needs at hand, the developers leverage the knowledge about the API aspects expressed in the opinions about the competing APIs.
   a) the DOCUMENTATION[11,11] support, e.g., when the official documentation is not sufficient enough, *"possibility to get an answer to a specific question (which may not be explained in other sources like the API's documentation)"*.
   b) the COMMUNITY[11,9] engagement in the forum posts and mailing lists, *"for instance, Firebase team members are active on Stack Overflow."*
   c) the USABILITY[1,1] and design principles of the API and the PERFORMANCE[8,8] of the API to assess API quality, *"It allows me to see common gotchas, so I can estimate the quality of the API based on those opinions, not just my own."*
4) To improve PRODUCTIVITY[9,8] by saving time in the decision making process, *"Getting experience of others can save a lot of time if you end up using a better API or skipping a bad one."*
5) To TRUST[13,11] and to *validate* the claims about a specific API usage or feature (e.g., if a posted code example is good/safe to use), because *"They represent hands-on information from fellow devs, free of marketing and business."*

Q18 *We asked the developers about the specific development needs that may motivate them to seek opinions about APIs.* We solicited opinions using a five-point Likert-scale question. The analysis of the responses (Figure 8) shows that developers find selection-related factors (i.e., determining replacement of an API and selection of an API among choices) to be the most influential for seeking for opinions (88% and 80.7% of all respondents agreed/strongly agreed, respectively). Developers also seek opinions when they need to improve a software feature or to fix a bug (69.9% and 68.7% agreement, respectively). 56.6% of the respondents seek help during the development of a new API, such as, addressing the shortcomings of the existing API. Developers are less enthusiastic to rely on opinions for validating their choice of an API to others (38.6%), or replacing one version of an API with another version (27.7%). Only 15.7% of the respondents mentioned that they seek opinions for reasons not covered by the above options. Note that while the 'Neutral' responses are not shown in Figure 8, we include the neutral counts in our calculation of percent agreements above. Our charts to show the results of Likert-scale questions (e.g., Figure 8) follow similar formats as adopted in previous research [103], [104].

Reasons for Seeking Opinions about APIs (RQ1.1)
Factors Motivating Opinion Seeking (RQ1.1.b)

Developers seek opinions about APIs in developer forums to support diverse development needs, such as building *expertise* about API aspects, learning nuances about specific API usage, making a selection of an API among choices, etc.

6.1.3 Challenges in Opinion Seeking (RQ1.1.c)

We asked developers two questions (Q5, 6 in Table 6) to determine the way developers seek information and opinions about APIs and the challenges they face while seeking and analyzing the opinions.

Q5 *We asked the developers to write about how they seek information about APIs in a developer forums and how they navigate through multiple forums posts to form an informed insight about an API.* More than 83% of the respondents include some form of SEARCHING[94,79] using a general purpose search engine (e.g., Google) or using the provided search capabilities in the forums. In the absence of a suitable alternative, the developers learn to trust such resources, *"I use Google and trust Stack Overflow's mechanisms for getting the best posts into the top search results"* However, they also cite that they learn to be patient to get the *right* results from the search engines, *"Google and patience"*. Because such search can still return many results, developers employ different mechanisms to navigate through the results and to find the information that they can consider as the *right* result. Such mechanisms include, the ranking (e.g., top hits) of the results, the manual SIMILARITY[3,3] assessment of the search results by focusing on the *recency* of the opinion, the analysis of the presence of SENTIMENTS[4,4] in the post about the API, *"I usually google keyword and then look for the positive and negative response"*.

Q6 *We asked developers to write about the challenges they face, while seeking for opinions about APIs from developer forums.* The major challenge developers reported was to get the SITUATIONAL RELEVANCY[26,22] of the opinions within their usage needs. Such difficulties can stem from diverse sources, such as, finding the right question for their problem, or realizing that the post may have only the fraction of the answers that the developer was looking for, *"It's hard to know what issues people have with APIs so it's difficult to come up with something to search for."* Another relevant category was RECENCY[13,13] of the provided opinion, as developers tend to struggle whether the provided opinion is still valid within the features offered by an API, *"Information may be outdated, applying to years-old API versions"*. The assessment of the TRUSTWORTHINESS[21,20] of the provided opinion as well as the opinion provider were considered as challenging due to various reasons, such as, lack of insight into the expertise level of the opinion provider, bias of opinion provider, etc. According to one developer, *"Sometimes it's hard to determine a developer's experience with certain technologies, and thus, it may be hard to judge the API based on that dev's opinion alone"*.

The INFORMATION OVERLOAD[3,3] in the forum posts while navigating through the forum posts was mentioned as a challenge, *"They provide many different opinions, and I have to aggregate all information and make my own decision"*. The absence of answers to questions, as well as the presence of low quality questions and answers, and the LACK OF INFORMATION[1,1] in the answers were frustrating to the developers during the analysis of the opinions. The developers mention about their extensive reliance on the

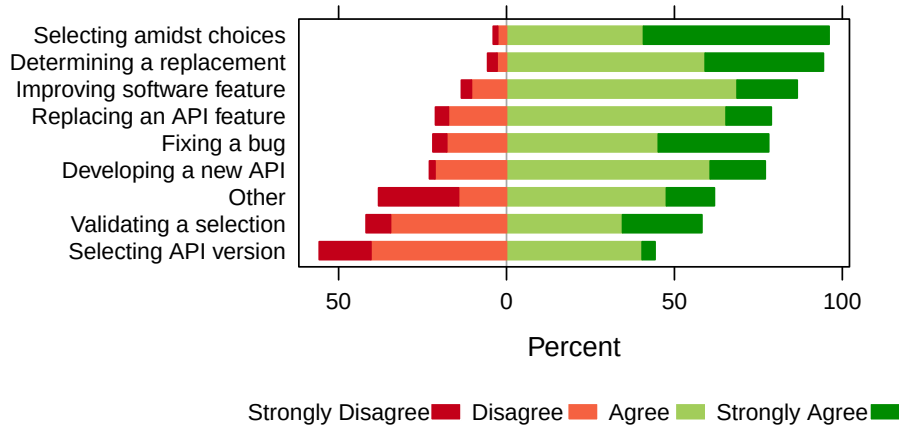

Fig. 8: Developer motivation for seeking opinions about APIs.

SEARCH[15,11] features offered by both Stack Overflow and general purpose search engines to find the right information. Finding the right keywords was a challenge during the searching. Developers wished for better search support to easily find duplicate and unanswered questions. Lack of knowledge on how to search can prove a blocker in such situations, *"If I'm inexperienced or lacking information in a particular area, I may not be using the right terminology to hit the answers I'm looking for."*

Developers expressed their difficulties during their SELECTION[6,6] of an API by leveraging the opinions. The lack of opinions and information for newer APIs is a challenge during the selection of an API, *"With newer APIs, there's often very little information."*. Developers can have specific requirements (e.g., performance) for their USAGE[8,8] of an API, and finding the opinion corresponding to the requirement can be non-trivial (e.g., is the feature offered by this API scalable?). The necessity to analyze opinions based on specific API aspects was highlighted by the developers, such as,

1) link to the DOCUMENTATION[4,4] support in the opinions,*"Ensuring that the information is up-to-date and accurate requires going back-and-forth between forums and software project's official docs to ensure accuracy (in this case official docs would be lacking, hence using forums is the first place)"*
2) COMPATIBILITY[4,4] of an API feature in different versions, *"filtering for a particular (current) API version is difficult."*
3) USABILITY[2,2] of the API, and the activity, and engagement of the supporting COMMUNITY[9,7], *"But looking at several questions in a particular tag will help give a feeling for the library and sometimes its community too."*

The lack of a proper mechanism to support such analysis within the current search engine or developer forums make such analysis difficult for the developers. Finally, getting an instant insight into the EXPERTISE[3,3] of the opinion provider is considered as important by the developers during their analysis of the opinions. The developers consider that getting such insight can be challenging, *"Need to figure out if the person knows what they are talking about and is in a situation comparable to ours."*

Reasons for Seeking Opinions about APIs (RQ1.1)
Challenges in Opinion Seeking (RQ1.1.c)

Majority of the developers leverage search engines to look for opinions. They manually analyze the presence of sentiments in the posts to pick the important information. The developers face challenges in such exploration for a variety of reasons, such as the difficulty in associating such opinions within the contexts of their development needs, the lack of enough evidence to validate the trustworthiness of a given claim in an opinion, etc.

6.2 Needs for API Review Quality Assessment (RQ1.2)

We asked three questions (Q3,7,24 in Table 6, two open-ended).

Q3 *We asked the developers whether they value the opinion about APIs of the other developers in the developer forums.* 89.2% of the respondents mentioned that they value the opinion of other developers in the online developer forums, such as, Stack Overflow. 10.8% reported that they do not value such opinions.

Q24 *We asked the 10.8% participants who do not value the opinion of other developers to provide us the reasons why they do not value such opinions.* The developers cited their concern about TRUSTWORTHINESS[4,3] as the main reason, followed by the lack of SITUATIONAL RELEVANCY[2,2] of the opinion to suit specific development needs. The concerns related to the trustworthiness of the opinion can stem from diverse factors, such as, the *credibility* and the *experience* of the poster, *"They are usually biased and sometimes blatant advertisements."*. Lack of trust can also stem from the inherent bias someone may possess towards a specific computing platform (e.g., Windows vs Linux developers). We note that both of these two categories were mentioned also by the developers who value opinions of other developers. However, they mentioned that by looking for diverse opinions about an entity, they can form a better insight into the trustworthiness of the provided claim and code example.

We now report the results from responses of the 89.2% developers who reported that they value the opinion of other developers.

Q7 *The open-ended question asked developers to write about the factors that determine the quality of the provided opinion.* The following factors are used to assess the quality of the opinions:

1) The quality of the provided opinions as comparable to API official DOCUMENTATION[54,49]. Forum posts are considered as informal documentation. The developers expected the provided opinions to be considered as an alternative source of API documentation. Clarity of the provided opinion with proper writing style and enough details with links to support each claim are considered as important when opinions can be considered as of good quality. In particular, the following metrics are cited to assess opinions as a form of API documentation.
   a) **Completeness** of the provided opinion based on the *"Exhaustiveness of the answer and comparison to others"*
   b) **Acceptable writing quality** by using *"proper language"* and *"Spelling, punctuation and grammar."*
   c) **Accompanying with facts**, such as using *screenshots or similar suggestions*, *examples from real world applications*, or *"Cross-referencing other answers, comments on the post, alternate answers to the same question."*
   d) **Brevity of the opinion** by being *"brief and to the point"*
   e) **Providing context** by discussing *"identified usage patterns and recognized API intent"*
   f) **Accompanying code examples with reaction**, *"Code samples, references to ISOs and other standards, clear writing that demonstrates a grasp of the subjects at hand"*
2) The REPUTATION[35,33] of the opinion provider and upvote to the forum post where the opinion is found, *"If the poster has a high Stack Overflow score and a couple of users comment positively about it, that weighs pretty heavily on my decision."*
3) The perceived EXPERTISE[13,12] of the opinion provider within the context of the provided opinion, *"It's easy to read a few lines of technical commentary and identify immediately whether the writer is a precise thinker who has an opinion worth reading."*
4) The TRUSTWORTHINESS[8,5] of the opinion provider who demonstrates *apparent fairness and weighing pros/cons*.
5) The SITUATIONAL RELEVANCE[12,9] of the provided opinion within a given development needs, and
6) The RECENCY[6,6] of the opinion, *"posts more than a few years old are likely to be out of date and counterproductive."*

> **Needs for API Review Quality Assessment (RQ1.2)**
>
> Developer analyze the quality of the opinions about APIs in the forum posts by considering the opinions as a source of *API documentation*. They employ a number of *metrics* to judge the quality of the provided opinions, e.g., the clarity and completeness of the provided opinion, the presence of code examples, the presence of detailed links supporting the claims, etc.

6.3 Tool Support for API Review Analysis (RQ2.1)

We asked four questions (Q8-10, 19 in Table 6, one open-ended) .

Q8 *We asked developers whether they currently rely on tools to analyze opinions about APIs from developer forums.* 13.3% responded with a 'Yes' and the rest (86.7%) with a 'No'.

Q9 *We further probed the developers who responded with a 'No'. We asked them whether they feel the necessity of a tool to help them to analyze those opinions.* The majority (62.5%) of the developers were unsure ('I don't know'). 9.7% responded with a 'Yes' and 27.8% with a 'No'.

Q10 *We further probed the developers who responded with a 'Yes'. We asked them to write the name of the tool they currently use.* The developers cited the following tools: 1) Google search, 2) Stack Overflow votes, 3) Stack Overflow mobile app, and 4) GitHub pulse.

Q19 was a multiple choice question, with each choice referring to one specific tool. The choice of the tools was inspired by research on sentiment analysis in other domains [77]. To ensure that the participants understood what we meant by each tool, we provided a one-line description to each choice. The choices were as follows (in the order we placed those in the survey):

1) **Opinion miner**: for an API name, it provides only the positive and negative opinions collected from the forums.
2) **Sentiment analyzer**: it automatically highlights the positive and negative opinions about an API in the forum posts.
3) **Opinion summarizer**: for an API name, it provides only a summarized version of the opinions from the forums.
4) **API comparator**: it compares two APIs based on opinions.
5) **Trend analyzer**: it shows sentiment trends towards an API.
6) **Competing APIs**: it finds APIs co-mentioned positively or negatively along an API of interest and compares those.

The respondents' first choice was the 'Competing APIs' tool (6), followed by a 'Trend Analyzer' to visualize trends of opinions about an API (5), an 'Opinion Miner' (1) and a 'Summarizer' (3) (see Figure 9). The sentiment analyzer (2) is the least desired tool, i.e., developers wanted tools that do not simply show sentiments, but also show the context of the provided opinions. Note that an opinion search and summarization engine in other domain (e.g, camera reviews) not only show the mined opinions, but also offer insights by summarizing and revealing trends. The engines facilitate the comparison among the competing entities through different aspects of the entities. Such aspects are also used to show the summarized viewpoints about the entities (ref Figure 3). Therefore, it was encouraging to see that developers largely agreed with the diverse summarization needs of API reviews and usage information, which corroborate with the similar summarization needs in other domains.

> **Tool Support for API Review Analysis (RQ2.1)**
>
> To analyze opinions about APIs in the forums, developers leverage traditional search features in the absence of a specialized tool. The developers agree that a variety of opinion analysis tools can be useful in their exploration of opinions about APIs, such as, opinion miner and summarizer, a trend analyzer, and an API comparator engine that can leverage those opinions to facilitate the comparison between APIs.

6.4 Needs for API Review Summarization (RQ2.2)

As we observed in the previous section, a potential opinion summarization engine about APIs need to show both summarized viewpoints about an API and support the comparison of APIs based on those viewpoints. To properly understand how and whether such summarization can indeed help developers, we probed the participants with a number of question. Specifically, we present the analysis of their responses along three themes (RQs 2.2a, b, and c in Sections 6.4.1-6.4.3.

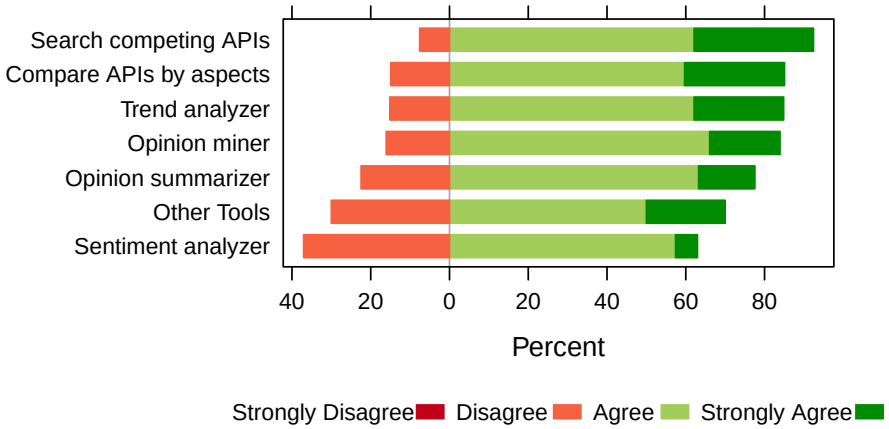

Fig. 9: Tool support for analyzing API opinions.

6.4.1 Factors Motivating Summarization Needs (RQ2.2.a)

We asked the developers three questions (Q20-22 in Table 6).

Q20 *We asked developers how often they feel overwhelmed due to the abundance of opinions about APIs in the developer forums.* 3.6% of the developers mentioned that they are 'Always' overwhelmed, 48.2% are 'Sometimes' overwhelmed and 37.3% are 'Rarely' overwhelmed. Only 10.8% developers mentioned that they are 'Never' overwhelmed by the abundance of opinions about APIs in the developer forums.

Q21 *We then asked them whether a summarization of those opinions would help them make better decisions about APIs?* 38.6% responded with a 'Yes' and 13.3% responded with a 'No'. 48.2% were unsure ('I don't know').

Q22 *We examined the needs to opinion summarization using a five-point Likert-scale with seven options*[5]: 1) Too many forum posts with opinions, 2) Opinions can evolve over time in different posts, 3) Opinions can change over time, 4) The interesting opinion may not be in the posts that you have looked in, 5) Contradictory opinions about an API may be missed, 6) Not enough time to look at all opinions, and 7) Other reason. The same options were also used in the pilot survey and we observed agreement from the developers across the options. We remove the analysis of two options ("Opinions can evolve over time in different posts" and "Opinions can change over time"), because both may be perceived as similar by the participants without a clarification. The analysis of the rest of the options (Figure 10) shows that nearly all developers agree that opinion summarization is much needed for several reasons. The vast majority of the respondents think that an interesting opinion about an API might be expressed in a post that they have missed or have not looked at. The need for opinion summarization is also motivated by the presence of too many posts to explore. Developers also believed that the lack of time to search for all possible opinions and the possibility of missing an important opinion are strong incentives for having opinions organized in a better way.

5. Before conducting the two surveys, the first author manually analyzed around 1,000 posts from Stack Overflow. The goal was to understand the type of opinions developers share about APIs in the forum posts. The options were selected based on our observation of API reviews in forum posts.

Needs for API Review Summarization (RQ2.2)
Factors Motivating Summarization Needs (RQ2.2.a)

The developers reported that they feel overwhelmed due to the abundance of opinions about APIs in the developer forums. The developers were in agreement that such difficulty arises due to a variety of reasons, such as, relevant opinions about an API may be missed because those opinions were in posts not checked by the developer, the lack of enough time to find all such opinions, etc.

6.4.2 Summarization Preferences (RQ2.2.b)

We asked three questions (two open-ended) to understand the relative benefits and problems developers may encounter during their usage of API review summaries from developer forums (Q13,14, and 23 in Table 6).

Q13 *We asked developers to write about the areas that can be positively impacted by the summarization of opinions about APIs from developer forums.* The following areas were considered to be benefited from the summaries:

1) The majority of the developers considered that API SELECTION[30,25] is an area that can reap the benefits from the summaries, *"It would make the initial review and whittling down of candidates quicker and easier."*. They also considered that API review summarization can improve the PRODUCTIVITY[12,11] of the developers by offering quick but informed insights about APIs, *"It would make the initial review and whittling down of candidates quicker and easier."*
2) The developers expected the summaries to assist in their API USAGE[7,7], such as, by showing reactions from other developers for a given code example that can offer SITUATIONALLY RELEVANT[3,3] insights about the code example (e.g., if the code example does indeed work and solve the need as claimed). The developers consider that they can use a summary as a form of documentation to explore the relative strengths and weaknesses of an API for a given development need. The developers do not expect the official documentation of an API to have such insights, because *"You can see how the code really works, as opposed to how the documentation thinks it works"*. Indeed, official documentation can be often incomplete [7]. Carroll et

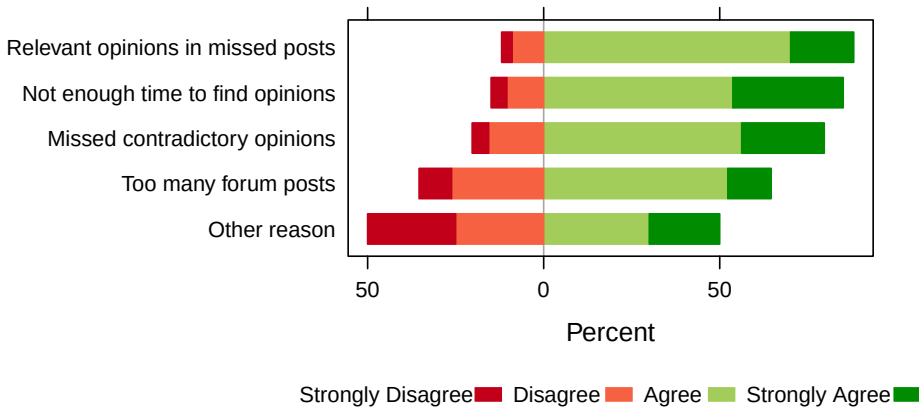

Fig. 10: Developer needs for opinion summarization about APIs.

al. [24] advocated the needs for *minimal API documentation*, where API usage scenarios should be shown in a task-based view. In a way, our developers in the survey expect the API usage scenarios combined with the opinions posted in the forum posts as effective ingredients to their completion of the task at hand. Hence, they expect that the summaries of opinions about API can help them with the efficient usage of an API during their development tasks, *"If summarization is beneficial to understanding API, then any problem even remotely related to that use stands to achieve a network benefit."*

3) As we noted earlier in this section, developers leverage the opinions about diverse API aspects to compare APIs during their selection of an API among choices. In the absence of any such automatic categorization available to automatically show how an API operates based on a given aspect (e.g., performance), the developers had expectations that an opinion summarizer would help them with such information. For example, the developers considered that opinion summaries can also improve the analysis about different API aspects, such as, USABILITY[2,2], COMMUNITY[2,2], PERFORMANCE[3,3], DOCUMENTATION[6,5], etc.
4) Finally, developers envisioned that opinion summaries can improve the SEARCH[6,6] for APIs and help them find better EXPERTISE[4,4] to learn and use APIs.

Q14 *We next asked developers to write about the areas that can be negatively impacted by the summarization of opinions about APIs from developer forums.* The purpose was to be aware of any drawback that can arise due to the use of API opinion summaries.

1) Developers considered that opinion summaries may MISS NUANCES[18,12] in the API behavior that are subtle, may not be as frequent, but that could be *"key design choices or the meta-information . . . "*. The REASONING[7,6] about opinions can also suffer because of that.
2) While developers considered the selection of an API as an area that can be positively affected by opinion summaries in general, they also raised the concern that summaries may negatively impact API SELECTION[6,6] when more subtle API aspects are not as widely discussed as other aspects, because *"Popularity and fashion trends may mask more objective quality criteria "*. Summaries may become API BARRIER[3,3] when new APIs with less number of opinions are not ranked higher in the summaries, *"Just relying on the summarization for deciding for or against an API will probably not be enough and may lead to decisions that are less than optimal."*

Q23 *We asked developers about how opinion summaries can support their tasks and decision making process.* There were nine options. We used the same options we used to solicit the needs for opinions about APIs in Figure 8. This was due to the fact that the purpose of an opinion summarizer should be to facilitate the easier and efficient consumption of opinions. In Figure 11, we present the responses of the developers. More than 70% of the developers believe that opinion summarization can help them with two decisions: 1) determining a replacement of an API and 2) selecting the right API among choices. Developers agree that summaries can also help them improve a software feature, replace an API feature, and validate the choice of API. Fixing a bug, while receiving 38.6% of the agreement, might not be well supported by summaries since this task requires certain detailed information about an API that may not be present in the opinions (e.g., its version, the code itself, the feature). The developers mostly disagreed with the use of opinions to select an API version (28.9% agreement only).

Needs for API Review Summarization (RQ2.2)
Summarization Preferences (RQ2.2.b)

The summarization of opinions about APIs can be effective to support the selection of an API among choices, the learning of API usage cases, etc. Developers expect a gain in productivity in supporting such needs by the opinion summarizer. However, developers expressed their cautions with a potential opinion summarizer for APIs, when for example, subtle nuances of an API can be missed in the summary. Another concern is that summaries can make it difficult for a new API to be adopted. For example, developers may keep using the existing APIs, because they are possibly reviewed more than the new APIs.

6.4.3 Summarization Types (RQ2.2.c)

We asked developers three questions (Q11,12,15 in Table 6) to learn about their preferences on how they would like to have opinion summaries about APIs to be produced.

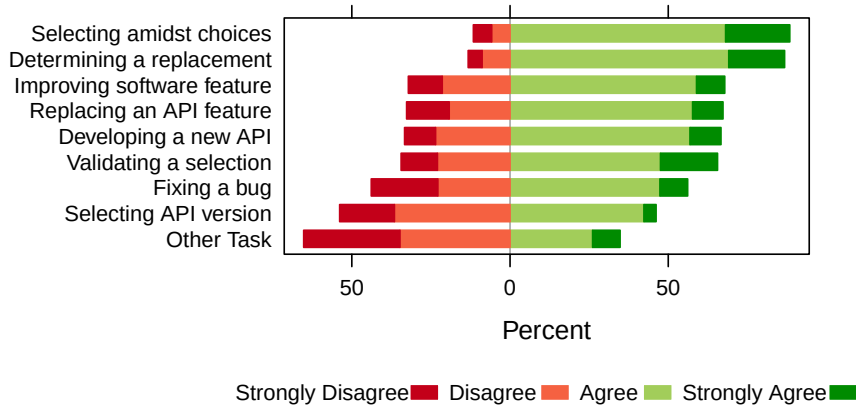

Fig. 11: Impact of opinion summaries on decision making about APIs.

Q11 *We asked developers to write about the different factors in an API that play a role during their decision to choose the API.* The purpose was to learn what specific API aspects developers consider as important. Intuitively, developers would like to see opinions about those aspects in the forum posts to make a decision on the API. The developers mentioned about the following API aspects as factors contributing to their decisions to choose an API:

1) **USABILITY**[87,56]
   a) Simple. *"Simplicity, compliance with standards"*
   b) Ease of use. *". . . easy to replace, simple to use"*
   c) Design. *"clearly designed & documented usage protocol"*
   d) Stable. *". . . , so it doesn't change every 10 days?"*
2) **DOCUMENTATION**[47,43]
   a) Clear. *"Clear documentation . . . "*,
   b) Consistent. *". . . , has to be intuitive (consistent naming)"*
   c) Complete and correct with examples. *"Correctness, completeness, documentation & examples"*
   d) Availability. *"clear and complete description, and a good blogging is a big bonus"*
3) **COMMUNITY**[31,22]
   a) Active Community. *"Code quality, maintainer activity, ratio of answered/unanswered questions about the API."*
   b) Mature. *"Strong Stack Overflow community (to show that the API is relatively mature)."*
   c) Reputation. *"Reputation of the organization."*, *"Other works of its creators."*
4) **PERFORMANCE**[30,23] *"Has it been used in production grade software before?"*
5) **COMPATIBILITY**[18,14]
   a) Language. *"Implemented in the language I need it."*
   b) Other API. *"Uniformity with other API . . . "*
   c) Framework/Project. *"compatibility (will it work for project I'm working on)"*
6) **LEGAL**[15,14] *". . . the API is open source and maintained."*
7) **PORTABILITY**[5,5] *"Thread safety, sync/async, cross-language and cross-OS support, etc. "*
8) **SECURITY**[2,2] *"Are security issues addressed quickly?"*
9) **BUG**[1,1] *"Features, ease of use, documentation, size and culture of community, bugs, quality of implementation, performance, how well I understand its purpose, whether it's a do-all framework or a targeted library, . . . "*

The USAGE SCENARIOS[10,9] of an API, and the FEATURES[4,4] it supports are also considered to be important factors to be part of the opinion summaries.

Q12 *We asked developers to provide their analysis on how opinions about APIs can be summarized when such opinions are scattered across multiple forum posts.* The purpose was to know what techniques developers consider as valuable to aggregate such opinions from diverse posts. While developers mention about using the SEARCH[12,12] features in Stack overflow, they acknowledged that they never thought of this problem deeply. They acknowledge that search engines or Stack Overflow features are really not the summarizer they may need, *"Often there is a sidebar with similar questions/discussions, but no easy way to "summarize"."*

To address their needs for such a summarizer, the idea of a dedicated API OPINION PORTAL[18,15] was discussed. In such a portal all opinions about an API can be aggregated and presented in one place, *"aggregate similar information, collect 'A vs B' discussions " . . . "like an aggregated portal about an API with all organized or non-organized pages"*. Having a centralized portal for such information can be useful, because *"Then I can read through them all and synthesize a solution."*. Developers advocated for machine learning approaches to develop such a portal *"It would be very interesting for this to be automated via machine learning."* Developers wished diverse presentation of opinions in the portal:

1) CATEGORIZED[8,8] into different API aspects, such as *"Summarization by different criteria, categorization"*,
2) Distributed by SENTIMENTS[8,5], such as, star rating (similar to other domains), *"do an x-out-of-5 rating for portability, stability, versatility and other attributes"*.
3) Grouped by CONTRASTIVE[7,7] viewpoints to learn the strengths and weaknesses of an API feature from multiple forum posts, *"I make a research if the there are contradicting opinions on stack overflow / etc."*
4) Combined with code examples as a form of API USAGE SCENARIOS[7,6], as well as integrated with the API FORMAL DOCUMENTATION[11,9].

According to one developer *"Unified documentation on Stack Overflow (beta) looks like a positive step in the right direction."* Unfortunately, the Stack Overflow documentation site was shut

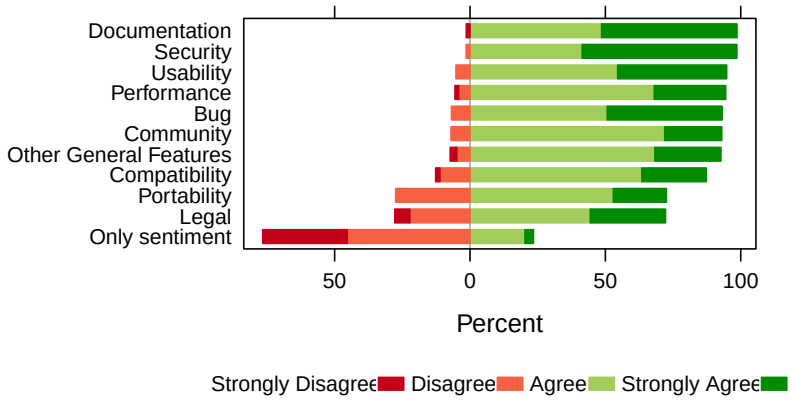

Fig. 12: Developers' preference to see opinions about API aspects.

down on August 8, 2017 [105], mainly due to a lower than the expected number of visits to the documentation site. Therefore, we can draw insights both from the problems faced by the Stack Overflow documentation site and from the responses of the developers in our survey to develop an API portal that can offer better user experiences to the developers.

Q15 *We asked developers about 11 API aspects, whether they would like to explore opinions about APIs around those aspects.* The 11 aspects are: 1) Performance, 2) Usability, 3) Portability, 4) Compatibility, 5) Security, 6) Bug, 7) Community, 8) Documentation, 9) Legal, 10) General Features, 11) Only sentiment. We note that each of these options are found in the open-ended questions already. Thus, the response to this question can be used to further ensure that we understood the responses of the developers. The results of the relevant Likert-scale survey question are summarized in Figure 12. The vast majority of the developers agrees that the presence of documentation, discussion of API's security features, mention of any known bugs in an API, its compatibility, performance, usability, and the supporting user community are the main attributes that contribute to the opinion quality. Developers could not agree whether a useful opinion should include a mention of any legal terms; while they agree that posts containing only one's sentiment (e.g., "I love API X" or "API Y sucks") without any rationale are not very useful. We note that there was a similar themed open-ended question in Q11. However, the developers were not shown Q15 before their response of Q11. Moreover, the two questions were placed in two separate sections, Q11 in section 4 and Q15 in section 6. There was only one question (i.e., Q11) in section 4. Therefore, the participants did not see Q15 or the options in Q15 during their response of Q11.

Needs for API Review Summarization (RQ2.2)
Summarization Types (RQ2.2.c)

Developers expect to see opinions about diverse API aspects, such as, usability, performance, security, compatibility, portability, etc. Developers wished for a dedicated opinion portal engine for APIs where they can search for an API and see all the opinions collected about the APIs from the developer forums.

7 Discussions

In this section, we summarize the key points from our primary survey with regards to the two research questions that we set forth to address (Section 7.1). We then analyze the findings from the two surveys along three demographics by: 1) profession in Section 7.2, 2) experience of the survey participants in Section 7.3, and 3) the nine programming languages used to sample the primary survey participants in Section 7.4.

7.1 Summary

In Table 9 and Table 10, we summarize the responses from our primary survey for RQ1 and RQ2, respectively. The insights are calculated using the same rules that we used in Section 4.

Observation 1. **Developers leverage opinions about APIs to support development needs, such as API selection, usage, learning about edge cases, etc (Table 9.) Developers expect opinion summaries can facilitate those needs by offering an increase in productivity, e.g., save time by offering quick but synthesized insights (Table 10)** In the absence of a specific opinion summarization portal available for APIs, though, we find that developers were unsure whether such summaries can be feasible. However, they recommend to leverage machine learning techniques to facilitate an opinion portal for APIs.

Observation 2. A number of questions were paired as open-ended and closed questions, with the open-ended questions being asked before the corresponding closed-end question. For example, we asked developers about the reasons behind why they seek opinions about APIs using two questions (Q4 and Q18). Another pair was Q11 and Q15, which we used to ask developers about their preference of specific API aspects/factors that they expect to see in the opinions about APIs.

In the first pair (i.e., Q4 and Q18) the responses to the open-ended question (Q4) show a slight variation from the response to the closed question (Q18). In Q4, the main reason mentioned was to build and look for expertise about APIs while seeking for opinions. Q18 does not have that option. Nevertheless, the majority of agreement for the options in Q18 show that those needs are also prevalent. In the second pair (i.e., Q11 and Q15), the responses to both questions produced an almost similar set of API aspects, such as, performance, usability, documentation, etc.

Observation 3. In both surveys, two closed questions were

TABLE 9: Highlights of findings from Primary survey about developers' needs to seek opinions about APIs (RQ1). Subscript with a question number shows number of responses.

RQ1.1	**Needs for seeking opinions about APIs from developer forums**
1_{114}	Do you visit developer forums to seek info about APIs? ***1) Yes 79.8%, 2) No 21.2%***
2_{93}	List top developer forums you visited in the last two years ***1) Stack Overflow & Exchange 56%, 2) Blogs & mailing lists 18.1%, 3) GitHub 5.4% 4) Google dev 4.8%, 5) Apple dev 3.6%***
4_{83}	What are your reasons for referring to opinions of other developers about APIs in online developer forums? ***1) Expertise 25.6%, 2) API usage 14%, 3) API selection 10.7%, 4) Opinion trustworthiness 10.7%, 5) Documentation/Community 9.1%***
5_{83}	How do you seek information about APIs in a developer forum? ***1) Search 83.9%, 2) Sentiment statistics 3.6% 3) Similarity analysis 2.7%, 4) Expertise 1.8%, 5) Recency/Documentation 0.9%***
6_{83}	What are your biggest challenges when seeking for opinions about an API in an online developer forum? ***1) Situational relevance 20.3%, 2) Trustworthiness 16.4% 3) Search 11.7%, 4) Recency 10.2%, 5) Community engagement 7%***
16_{83}	Where do you seek help/opinions about APIs? ***1) Stack Overflow 34.7%, 2) Co-workers 26.4%, 3) Internal mailing lists 10%, 4) IRC Chats 9.2%, 5) Search/Diverse sources 19.7%***
17_{83}	How often do you refer to developer forums to get information about APIs? ***1) Every day 36.1%, 2) Two/three times a week 32.5%, 3) Once a month 16.9%, 4) Once a week 14.5%***
18_{83}	When do you seek opinions about APIs? ***1) Selection among choices 88%, 2) Determining replacement 80.7%, 3) Feature enhancement 69.9%, 4) Fixing a bug 68.7%, 5) Develop competing API 56.6%, 6) Replace a feature 49.4%, 7) Validate a selection 38.6%, 8) Select API version 27.7%, 9) Other task 15.7%***
RQ1.2	**Needs for opinion quality assessment**
3_{93}	Do you value the opinion of other developers in the developer forums when deciding on what API to use? ***1) Yes 89.2%, 2) No 10.8%***
7_{83}	What factors in a forum post can help you determine the quality of a provided opinion about an API? ***1) Documentation 35.5%, 2) Reputation 23%, 3) Expertise 8.6%, 4) Situational relevance/Community 7.9%, 5) Trustworthiness 5.3%***
24_{10}	Please explain why you don't value the opinion of other developers in the developer forums. ***1) Trustworthiness 23.5% 2) Situational relevance 11.8%, 3) Community 11.8%, 4) Expertise 11.8%,***

paired together (Q18 and Q23 in the primary survey and Q4 and Q12 in the pilot survey). The first question in each pair (i.e., Q18 in primary and Q4 in pilot surveys) aims at understanding the needs of developers for seeking opinions about APIs. The second question in each pair (i.e., Q23 in primary survey and Q12 in pilot one) aims to understand the needs for summarizing such opinions. The options for each of the four questions remained the same (see Table 9). In both surveys, the highest ranked option was "selection among choices". Therefore, developers seek opinions to decide on an API among choices and they believe that the summarization of opinions can assist them in their selection process. The three selection-related options (i.e., selection among choices, determining a replacement, and validating an API selection) are ranked as the top three in the pair of questions from the primary survey. **Therefore, the primary focus to aggregate and summarize opinions about APIs would be to assist developers in their selection of APIs and any other tasks relevant to it.**

7.2 Analysis by Professions

In Tables 11 and 12, we summarize the results of closed questions from our final survey by the reported professions of survey participants. We report each question as follows:

1) Multiple Choice Questions. If the question has three options (Yes, No, I don't know), we only show the result of the option(s) with the majority agreement.
2) Likert-scale Questions. We show all the options, by calculating agreements using the formula we introduce in Section 4 (we used in Tables 5,9, and 10).
3) We highlight the value of an option for each profession as *bold italic*, if the option has the maximum value out of all options for the profession. For example, for Q16 (where do you seek opinions about APIs?) in Table 11, there are five options: a) Internal mailing lists, b) IRC chats, c) Co-workers, d) Stack Overflow, e) Search/Diverse sources The software developers in our survey show the most agreement (89.2%) towards Stack Overflow. We thus make the value bold-italic among the five option for software developers.
4) We place a star symbol beside a highlighted value of an option, if the value is statistically significant[6]

Observation 4. **The necessity of seeking opinions about APIs is mostly prevalent and consistent across the different reported professions in our surveys.** All the research engineers, students, and team leads, and 89.2% of software developers who visit developer forums in our survey, also value the opinions about APIs in the forums. Unlike managers, technical leads are expected to work closely with the codebase and overall system architecture design. The decision on an API during the design of a system can be beneficial for such team leads. According to one team lead: *"The quality of APIs can vary considerably. Getting experience of others can save a lot of time if you end up using a better API or end up skipping a bad one."* For the researchers, the motivation was to learn from the experts: *" they have used the API, probably more extensively then I have, and may be experts in their subfield"*.

Observation 5. While all the team leads consult API information in the developer forums, most of them visit the forum two or three times a week. In contrast, most of the developers who consult developer forums for API information, do so more every day.

Observation 6. In both pilot and primary surveys, we asked developers about their preference for tools to better support their

6. We use Mann Whitney U test and a 95% confidence level (i.e., $p = 0.05$).

TABLE 10: Highlights of findings from Primary survey about developers' needs for tool support to analyze opinions about APIs (RQ2). Subscript with a question number shows number of responses.

RQ2.1	**Tool support**
8_{83}	Do you rely on tools to help you understand opinions about APIs in online forum discussions? ***1) Yes 13.3%, 2) No 86.7%***
9_{72}	If you don't use a tool currently to explore the diverse opinions about APIs in developer forums, do you believe there is a need of such a tool to help you find the right viewpoints about an API quickly? ***1) Yes 9.7%, 2) No 27.8%, 3) I don't know 62.5%***
10_{11}	You said yes to the previous question on using a tool to navigate forum posts. Please provide the name of the tool ***1) Google/Search 4, 2) Stack Overflow votes/app/related post 3, 3) GitHub issue/pulse 1, 4) Safari 1, 5) Reference documentation 1***
19_{83}	What tools can better support your understanding of API reviews in developer forums? ***1) Opinion mining 56.6%, 2) Sentiment analysis 41%, 3) Opinion summarization 45.8%, 4) API comparator 68.7%, 5) Trend analyzer 67.5%, 6) Co-mentioned competing APIs 73.5%, 7) Other Tools 8.4%***
RQ2.2	**Needs for Opinion Summarization**
11_{83}	What are the important factors of an API that play a role in your decision to choose an API? ***1) Usability 30.1%, 2) Documentation 16.3%, 3) Community 10.7%, 4) Performance 10.4%, 5) Compatibility 6.2%, 6) Legal 5.2%,***
12_{83}	Considering that opinions and diverse viewpoints about an API can be scattered in different posts and threads of a developer forum, what are the different ways opinions about APIs can be summarized from developer forums? ***1) Portal for opinion aggregation 12.7%, 2) Search 8.5% 3) Documentation 7.7%, 4) Sentiment statistics/Opinion categorization 5.6%, 5) Contrastive summaries/Usage scenario summaries 4.9%***
13_{83}	What areas can be positively affected by the summarization of reviews about APIs from developer forums? ***1) API selection 26.1%, 2) Productivity 10.4% 3) API usage 6.1% 4) Documentation 5.2% 5) API popularity analysis 3.5%***
14_{83}	What areas can be negatively affected by the summarization of reviews about APIs from developer forums? ***1) Opinion trustworthiness 19.3% 2) Missing nuances 16.5% 3) Opinion reasoning 6.4% 4) API selection 5.5% 5) New API Entry 2.8%***
15_{83}	An opinion is important if it contains discussion about the following API aspects? ***1) Usability 88% 2) Documentation 85.3% 3) Security 83.1% 4) Performance 81.9% 5) Bug 81.9% 6) Compatibility 66.3%, 7) Community 63.9%, 8) Legal 47%, 9) Portability 44.6%, 10) General API Features 45.8%, 11) Only Sentiment 18.1%***
20_{83}	How often do you feel overwhelmed due to the abundance of opinions about an API? ***1) Sometimes 48.2% 2) Rarely 37.3% 3) Never 10.8% 4) Always 3.6%***
21_{83}	Would a summarization of opinions about APIs help you to make a better decision on which one to use? ***1) I don't know 48.2% 2) Yes 38.6% 3) No 13.3%***
22_{83}	Opinions about an API need to be summarized because ***1) The interesting opinion may not be in the posts that you have looked in 71.1% 2) Not enough time to look at all opinions 68.7% 3) Contradictory opinions about an API may be missed 61.4% 4) Too many forum posts with opinions 50.6% 5) Other reason 12%***
23_{83}	Opinion summarization can improve the following decision making processes ***1) Selection among choices 73.5%, 2) Determining replacement of an API 71.1%, 3) Validating an API selection 48.2%, 4) Feature enhancement 45.8% , 5) Replacing an API feature 42.2%, 6) Fixing a bug 38.6%, 7) Selecting API version 28.9%, 8) Other 9.6%***

understanding of opinions about APIs in the developer forums (Q19 in primary survey, Q7 in pilot survey). A summarization engine to compare APIs based on different features is considered as the most useful among the software developers, team leads and research engineers. In our pilot survey, the leads also preferred the same tool the most among all tools. In our pilot survey, we got responses from three managers, who showed equal preference (33.3%) to all the tools except an opinion miner (66.7%).

Observation 7. There is a more clear distinction among the professions in their preference of API aspects that they prefer to explore in the opinions about APIs (Q15 in primary survey). The team leads are most interested to find opinions about API documentation, while the other professions including software engineers are most interested to learn about the usability of the API. All the professions agreed the most that the summarizing of opinions can help in the selection of APIs.

7.3 Analysis by Experiences

In Tables 13 and 14, we summarize the results of closed questions from our final survey by the reported experiences of survey participants following the same reporting principles we discussed in Section 7.2.

Observation 8. In both pilot and primary surveys, the developers with less experience show more interest to value the opinion of other developers (Q2 in Table 13). In both the surveys, the more experienced developers offer more uniform preferences towards the different tools that could be developed to facilitate opinion analysis from developer forums (Q19 in Table 14).

Observation 9. The less experienced developers visit forums more frequently (Q17 in primary survey): 75%, 61.2%, and 58.2% developers with 3-6, 7-10, and 10+ years of experience, respectively visit developer forums at least two or three times a week.

Observation 10. Among the less experienced developers (3-6 years of experience), both Trend Analyzer and API Comparator were ranked over other tools when asked about their preference of tools to better support their understanding of opinions about APIs (Q19 in primary survey). The developers with 10+ years of experience were most interested to explore the Competing APIs tool. The preference towards a specific tool decreases as developers become more experienced. For example, the developers with 10+ years of experience prefer the different tools with almost similar preference. The two tools (API Comparator and Competing APIs) are also ranked the highest (70.7%) by developers with 10+ years

TABLE 11: Highlights of Findings from Primary Survey RQ1 Closed Questions by Profession

No	Question
1	Do you visit developer forums to seek info about APIs? Research Engineer *1) Research Engineer:* ***Yes 100%****, 2) Student:* ***Yes 66.7%****, 3) Software Developer:* ***Yes 78.7%****, 4) Lead:* ***Yes 100%****
3	Do you value the opinion of other developers in the developer forums when deciding on what API to use? *1) Research Engineer:* ***Yes 100%****, 2) Student:* ***Yes 100%****, 3) Software Developer:* ***Yes 89.2%****, 4) Lead:* ***Yes 100%****

16 — Where do you seek help/opinions about APIs?

	Internal Mailing list	IRC chats	Co-workers	Stack Overflow	Search/Diverse Sources
Software Developer	24.3%	24.3%	71.6%	***89.2%****	39.2%
Other	-	-	-	***33.3%***	***33.3%***
Student	50.0%	50.0%	33.3%	***100.0%***	50.0%
Research Engineer	33.3%	33.3%	33.3%	***100%****	66.7%
Lead	36.4%	18.2%	63.6%	***100%****	63.6%

17 — How often do you refer to online forums (e.g., Stack Overflow) to get information about APIs?

	Every day	Two or three times a week	Once a week	One a month	Never
Software Developer	***33.3%****	12.2%	28.4%	14.9%	11.2%
Other	33.3%	0.0%	0.0%	0.0%	***66.7%***
Student	***50.0%***	0.0%	0.0%	***50.0%***	0.0%
Research Engineer	***33.3%***	***33.3%***	***33.3%***	0.0%	0.0%
Lead	27.3%	***36.4%****	18.2%	18.2%	0.0%

18 — When do you seek opinions about APIs?
SAC = Selecting amidst choices, DAR = Determining a replacement, ISF = Improving software feature, RAF = Replacing software feature, DAN = Developing a new API, FAB = Fixing a bug, SAV = Selecting API version, VAS = Validating a selection

	SAC	DAR	ISF	RAF	DAN	VAS	FAB	SAV	Other
Software Developer	***81.1%****	74.3%	60.8%	45.9%	51.4%	39.2%	59.5%	25.7%	17.6%
Other	0.0%	33.3%	***33.3%***	***33.3%***	***33.3%***	0.0%	***33.3%***	0.0%	0.0%
Student	***100.0%***	***100.0%***	50.0%	100.0%	100.0%	0.0%	50.0%	0.0%	0.0%
Research Engineer	66.7%	33.3%	66.7%	0.0%	66.7%	0.0%	***100.0%***	0.0%	0.0%
Lead	***81.8%****	72.7%	***81.8%****	36.4%	36.4%	27.3%	72.7%	36.4%	0.0%

of experience in the pilot survey (Q7).

Observation 11. The developers with 10+ years of experience show almost equal preference to the different implicit API aspects about which they prefer to see or seek opinions. The less experienced developers also have more specific preference of API aspects about which they like to explore the opinions of other developers (Q15 in primary survey): 1) More than 75% agreement for the six API aspects by developers with 3-6 years of experience (maximum 100% for Usability) 2) More than 75% agreement for three API aspects by developers with 7-10 years of experience (maximum 84.6% for both usability and documentation) 3) Maximum 72.7% agreement for two API aspects by developers with 10+ years of experience (Documentation and Bug).

7.4 Analysis by Programming Languages

In Tables 15 and 16, we summarize the results of closed questions from our final survey by the nine targeted programming languages from where our survey sample was collected. We follow the same reporting principles that we discussed in Section 7.2.

Observation 12. Among the nine programming languages that we targeted in our primary survey to sample the participants of our primary survey, at least 70% respondents from each language reported that they visit developer forums to seek information about APIs. The opinions posted about APIs in the forums are valued by those developers (100% of all Java developers, followed by at least 78% of the respondents from other languages).

Observation 13. All the Java developers also report that they do not have any tools that they can currently use to analyze those opinions. The lack of such tool support is also prevalent among developers across other languages.

Observation 14. The developers across all the programming languages (except Python and C#) preferred most the tools that can offer summarized comparisons between APIs or show list of competing APIs. For python, the most preferred tool was an opinion miner and for C# it was a trend analyzer.

Observation 15. While developers prefer to explore opinions about diverse API aspects, the usability aspect was ranked the highest among the list of API aspects across all the languages except C, C++ and Javascript. For C and C++, the most pressing aspects are documentation and for Javascript it was performance.

8 Research Journey

We noted in Section 1 (Figure 2) that the findings from the surveys led us to develop techniques and tools to assist developers in their analysis of opinions and usage about APIs from Stack Overflow. In this section, we break down the journey into two major phases that we undertook after the surveys:

1) Identification of core and actionable insights from the survey results that can be used to guide the design of tools to assist developers in their analysis of opinions and usage of APIs from developer forums (see Section 8.1).
2) Development and evaluation of techniques and our tool (Opiner) based on the insights (see Section 8.2).

8.1 Core and Actionable Findings

In Table 17, we summarize the core findings obtained from the surveys. Each finding is provided a unique ID (denoted by CID). The first two columns show the research questions, the third column presents the core findings. In Table 18, we identify requirements for future tool designs to support each core finding. Each requirement is mapped to the CIDs from Table 17 (first column). The second column shows the requirements, i.e., actionable findings. We cluster the requirements into five categories (R1-R5). The third column lists the questions from the primary survey

TABLE 12: Highlights of Findings from Primary Survey RQ2 Closed Questions by Profession

No	Question
8	Do you rely on tools to help you understand opinions about APIs in online forum discussions? *1) Research Engineer:* ***No 66.7%****, 2) Student:* ***No 100%****, 3) Software Developer:* ***No 79.7%****, *4) Lead:* ***No 72.7%****
9	If you don't use a tool currently to explore the diverse opinions about APIs in developer forums, do you believe there is a need of such a tool to help you find the right viewpoints about an API quickly? *1) Research Engineer:* ***No 33.3%, I don't know 33.3%*** *2) Student:* ***Yes 50%, I don't know 50%*** *3) Software Developer:* ***I don't know 48.6%****, *4) Lead:* ***I don't know 54.5%****

15 An opinion is important if it contains discussion about the following API aspects?
Per = Performance, Sec = Security, Use = Usability, Doc = Documentation, Comp = Compatibility, Comm = Community, Leg = Legal, Port = Portability, Sen = Only Sentiment, Feat = General Features

	Per	Sec	Use	Doc	Comp	Comm	Bug	Leg	Por	Sen	Feat
Software Developer	75.7%	74.3%	***82.4%****	75.7%	60.8%	56.8%	74.3%	41.9%	40.5%	13.5%	43.2%
Other	***33.3%***	***33.3%***	***33.3%***	0.0%	***33.3%***	0.0%	0.0%	0.0%	***33.3%***	0.0%	***33.3%***
Student	50.0%	***100.0%***	***100.0%***	***100.0%***	50.0%	50.0%	100.0%	0.0%	50.0%	50.0%	0.0%
Research Engineer	***66.7%***	***66.7%***	***66.7%***	***66.7%***	33.3%	***66.7%***	***66.7%***	***66.7%***	0.0%	33.3%	33.3%
Lead	72.7%	81.8%	63.6%	***100%****	63.6%	72.7%	81.8%	54.5%	45.5%	27.3%	36.4%

19 What tools can better support your understanding of API reviews in developer forums?
OM = Opinion Miner, SA = Sentiment Analyzer, OS = Opinion Summarizer, AC = API Comparator, TA = Trend Analyzer, CA = Competing APIs, OT = Other Tools

	OM	SA	OS	AC	TA	CA	OT
Software Developer	54.1%	39.2%	43.2%	62.2%	59.5%	***64.9%****	6.8%
Other	0.0%	0.0%	0.0%	***33.3%***	***33.3%***	***33.3%***	0.0%
Student	***100.0%***	0.0%	50.0%	50.0%	***100.0%***	***100.0%***	0.0%
Research Engineer	0.0%	33.3%	33.3%	***100.0%***	66.7%	66.7%	33.3%
Lead	45.5%	36.4%	36.4%	54.5%	63.6%	***72.7%****	9.1%

20 How often do you feel overwhelmed due to the abundance of opinions about an API?

	Always	Sometimes	Rarely	Never
Software Developer	4.1%	***45.9%****	31.1%	8.1%
Other	***33.3%***	0.0%	0.0%	0.0%
Student	0.0%	***100.0%***	0.0%	0.0%
Research Engineer	0.0%	***33.3%***	***33.3%***	***33.3%***
Lead	0.0%	27.3%	***63.6%****	9.1%

21	Would a summarization of opinions about APIs help you to make a better decision on which one to use? *1) Research Eng: No 66.7%, 2) Student: Yes 50%, don't know 50%, 3) Software Dev:* ***don't know 40.5%****, *4) Lead:* ***don't know 63.6%****

22 Opinions about an API need to be summarized because

	Too many posts	Opinion in missed posts	Missed contradictory	Not enough time	Other reason
Software Developer	50.0%	***63.5%****	54.1%	59.5%	12.2%
Other	***33.3%***	***33.3%***	***33.3%***	***33.3%***	0.0%
Student	50.0%	***100.0%***	***100.0%***	***100.0%***	0.0%
Research Engineer	0.0%	***66.7%***	***66.7%***	***66.7%***	33.3%
Lead	27.3%	63.6%	54.5%	***72.7%****	0.0%

23 Opinion summarization can improve the following decision making processes
SAC = Selecting amidst choices, DAR = Determining a replacement, ISF = Improving software feature, RAF = Replacing software feature, DAN = Developing a new API, FAB = Fixing a bug, SAV = Selecting API version, VAS = Validating a selection

	SAC	DAR	ISF	RAF	DAN	VAS	FAB	SAV	Other
Software Developer	***63.5%****	60.8%	41.9%	36.5%	37.8%	45.9%	33.8%	24.3%	10.8%
Other	***33.3%***	***33.3%***	0.0%	***33.3%***	0.0%	0.0%	0.0%	***33.3%***	0.0%
Student	***100.0%***	***100.0%***	0.0%	50.0%	0.0%	50.0%	50.0%	0.0%	0.0%
Research Engineer	***66.7%***	***66.7%***	***66.7%***	33.3%	***66.7%***	33.3%	33.3%	0.0%	0.0%
Lead	***81.8%****	***81.8%****	45.5%	45.5%	36.4%	36.4%	45.5%	45.5%	0.0%

that we used to identify those requirements. The last column presents the features implemented in our tool Opiner to address those requirements. In Figure 13, we show how requirements are implemented into our tool. We now discuss the findings below along with the elicited requirements.

8.1.1 Needs for Mining API Opinions & Usage (R1)

The developers in our surveys seek and analyze opinions about APIs to support diverse development needs and they consider the combination of opinions and code examples in the forum posts as a form of API documentation (CID 1, 2 in Table 17). To search for opinions and usage discussions about APIs, developers use search engines or the tags in Stack Overflow. They raise the concern that this approach of using search engines (e.g., find sentiments about an API) can be sub-optimal when they only explore the top results (CID 3 in Table 17). The developers reported to look for sentiments and situational relevance in the search results (Q5). This exploration can be challenging, because the results may not be (1) *situationally relevant*, such as the API about which they would like to see opinions about may not be present in the search result. It could also be that their development task is not properly supported by the code example found in the search result. (2) *trustworthy*, such as, sentiment towards an API found in the search result may not represent the overall sentiments expressed towards it (e.g., the opinion in the search results may be biased). (3) *recent*, such as the opinion and code example may not be the most recent and thus the solution may not be applicable to the most recent version of the API. In the absence of a dedicated engine for APIs, developers opt for the reformulation of their search query

TABLE 13: Highlights of Findings from Primary Survey RQ1 Closed Questions by Experience

No	Question
1	Do you visit developer forums to seek info about APIs? Research Engineer *1) 10+:* ***Yes 76.8%****, *2) 7-10:* ***Yes 83.9%****, *3) 3-6:* ***Yes 85.7%****
3	Do you value the opinion of other developers in the developer forums when deciding on what API to use? *1) 10+:* ***Yes 85.5%****, *2) 7-10:* ***Yes 92.3%****, *3) 3-6:* ***Yes 100%****
16	Where do you seek help/opinions about APIs?

	Internal Mailing list	IRC chats	Co-workers	Stack Overflow	Search/Diverse Sources
10+	21.8%	10.9%	67.3%	***85.5%****	41.8%
7–10	30.8%	34.6%	69.2%	***92.3%****	46.2%
3–6	33.3%	58.3%	66.7%	***100.0%****	41.7%

17	How often do you refer to online forums (e.g., Stack Overflow) to get information about APIs?

	Every day	Two or three times a week	Once a week	One a month	Never
10+	***32.7%****	25.5%	12.7%	14.5%	0.0%
7–10	***30.8%****	30.8%	11.5%	19.2%	7.7%
3–6	33.3%	***41.7%****	16.7%	8.3%	0.0%

18	When do you seek opinions about APIs? SAC = Selecting amidst choices, DAR = Determining a replacement, ISF = Improving software feature, RAF = Replacing software feature, DAN = Developing a new API, FAB = Fixing a bug, SAV = Selecting API version, VAS = Validating a selection

	SAC	DAR	ISF	RAF	DAN	VAS	FAB	SAV	Other
10+	***74.5%****	65.5%	60.0%	38.2%	45.5%	34.5%	47.3%	25.5%	10.9%
7–10	***80.8%****	84.6%	61.5%	53.8%	57.7%	38.5%	***80.8%****	26.9%	19.2%
3–6	***91.7%****	75.0%	75.0%	50.0%	58.3%	25.0%	83.3%	16.7%	16.7%

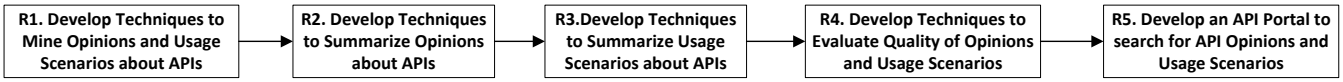

Fig. 13: The major research steps undertaken to incorporate the requirements from the two surveys into our proof of concept tool, Opiner

by modifying search keywords. This approach is considered as challenging and time consuming. The developers wished for better search support to address their needs.

Intuitively, it is easier to find opinions/solutions for an API with regards to specific task (or situation), if they are not *scattered* in millions of posts in the forums. As a first step towards facilitating such search, it is thus necessary to collect opinions and code examples about APIs from the millions of posts, where their usage is discussed. During our tool design to assist developers in their exploration of opinions and usage about APIs from developer forums, we formulate the following two requirements:

1) **Opinions about an API need to be collected**. Sentiment and emotion mining in software engineering has so far focused on the usefulness of cross-domain sentiment detection tools for software engineering, the development of sentiment detection tools for the domain of software engineering, and the relationship between team productivity with the sentiments expressed (see Section 2.2). We are aware of no technique that can mine opinions associated to an API from developer forums. We developed a framework to automatically mine opinions about APIs from developer forums. The framework currently supports the following major features: a) Detection of API names in the forum texts b) Detection of opinionated sentences in the forum texts, and c) Association of opinionated sentences to APIs. A detailed description and evaluation of the framework is the subject of our paper [78].
2) **Code examples with reactions need to be collected together**. A number of recent research efforts have been devoted to mine code examples about APIs from forums, such as detecting all the APIs used in a code example in a forum post [18], [64], etc. We are aware of no technique that mines both a code example and the reactions towards the code example for an API from forum post. We developed another framework to automatically mine usage scenarios about APIs from developer forums. Each usage scenario about an API consists of three major components: a) The code example b) A short description in natural language about the code example, and c) The reactions of developers towards the forum post from where the code example is found. A detailed description and evaluation of the framework is the subject of our technical report [106].

8.1.2 Needs for Summarization of API Opinions (R2)

In our primary survey, 89.2% of the developers mentioned that they are overwhelmed by the huge volume of opinions posted about APIs in forums (CID 7). While the majority of the developers agreed that opinion summaries can help them evaluating APIs, the following summarization types were rated as (could be) useful (CID 8): 1) categorization of opinions by API aspects, 2) show trends of API popularity, 3) show contrastive viewpoints, 4) show dashboards to compare APIs, 5) opinions summarized as topics; and 6) most importantly opinions summarized into a paragraph. It would be interesting to further investigate the relative benefit of each summarization type and compare such findings with other domains. For example, Lerman et al. [107] found no clear winner for consumer product summarization, while aspect-based summarization is predominantly present in camera and phone reviews [77], and topic-based summarization has been studied in many domains (e.g., identification of popular topics in bug report detection [108], software traceability [109], etc.).

TABLE 14: Highlights of Findings from Primary Survey RQ2 Closed Questions by Experience

No 8 — Do you rely on tools to help you understand opinions about APIs in online forum discussions?
1) 10+: ***No 70.9%****, 2) 7-10:* ***No 84.6%****, 3) 3-6:* ***No 91.7%****

No 9 — If you don't use a tool currently to explore the diverse opinions about APIs in developer forums, do you believe there is a need of such a tool to help you find the right viewpoints about an API quickly?
1) 10+: ***I don't know 45.5%****, 2) 7-10:* ***I don't know 53.8%****, 3) 3-6:* ***I don't know 50%****

No 15 — An opinion is important if it contains discussion about the following API aspects?
Per = Performance, Sec = Security, Use = Usability, Doc = Documentation, Comp = Compatibility, Comm = Community, Leg = Legal, Port = Portability, Sen = Only Sentiment, Feat = General Features

	Per	Sec	Use	Doc	Comp	Comm	Bug	Leg	Por	Sen	Feat
10+	69.1%	70.9%	70.9%	***72.7%****	52.7%	58.2%	***72.7%****	43.6%	41.8%	18.2%	38.2%
7–10	69.2%	76.9%	***84.6%****	***84.6%****	61.5%	50.0%	65.4%	42.3%	38.5%	19.2%	13.3%
3–6	100.0%	83.3%	***100.0%****	75.0%	83.3%	66.7%	91.7%	33.3%	33.3%	0.0%	50.0%

No 19 — What tools can better support your understanding of API reviews in developer forums?
OM = Opinion Miner, SA = Sentiment Analyzer, OS = Opinion Summarizer, AC = API Comparator, TA = Trend Analyzer, CA = Competing APIs, OT = Other Tools

	OM	SA	OS	AC	TA	CA	OT
10+	49.1%	40.0%	38.3%	54.5%	50.9%	***63.6%****	9.1%
7–10	50.0%	30.8%	42.3%	65.4%	***69.2%****	***69.2%****	7.7%
3–6	58.3%	33.3%	50.0%	***83.3%****	***83.3%****	66.7%	0.0%

No 20 — How often do you feel overwhelmed due to the abundance of opinions about an API?

	Always	Sometimes	Rarely	Never
10+	1.8%	34.5%	***40.0%****	9.1%
7–10	3.8%	***50.0%****	26.9%	11.5%
3–6	8.3%	***66.7%****	16.7%	8.3%

No 21 — Would a summarization of opinions about APIs help you to make a better decision on which one to use?
1) 10+: ***I don't know 43.6%****, 2) 7-10:* ***I don't know 46.2%****, 3) 3-6:* ***Yes 58.3%****

No 22 — Opinions about an API need to be summarized because

	Too many posts	Opinion in missed posts	Missed contradictory opinion	Not enough time	Other reason
10+	36.4%	***60.0%****	54.5%	58.2%	10.9%
7–10	50.0%	***65.4%****	53.8%	61.5%	11.5%
3–6	***75.0%****	***75.0%****	58.3%	***75.0%****	8.3%

No 23 — Opinion summarization can improve the following decision making processes
SAC = Selecting amidst choices, DAR = Determining a replacement, ISF = Improving software feature, RAF = Replacing software feature, DAN = Developing a new API, FAB = Fixing a bug, SAV = Selecting API version, VAS = Validating a selection

	SAC	DAR	ISF	RAF	DAN	VAS	FAB	SAV	Other
10+	61.8%	***63.6%****	41.8%	34.5%	40.0%	45.5%	30.9%	45.5%	5.5%
7–10	***65.4%****	61.5%	46.2%	38.5%	34.6%	46.2%	42.3%	30.8%	11.5%
3–6	***83.3%****	66.7%	25.0%	50.0%	25.0%	25.0%	33.3%	16.7%	16.7%

As a first step towards facilitating the summarization of opinions about APIs, we formulate the following design requirements:

1) **Categorize opinions by aspect.** The developers indicated that they would like to seek opinions about diverse API aspects, such as performance, usability, security, etc. We develop machine learning techniques to automatically categorize each opinionated sentences associated to an API into 11 different categories: (i) Performance (ii) Usability (iii) Security (iv) Compatibility (v) Portability (vi) Legal (vii) Bug (viii) Community (ix) Documentation (x) Other general features (xi) Only sentiment . The first nine categories are frequently asked for in the responses of our surveys (e.g., Q11, Q15 in our primary survey).
2) **Statistical summarization.** Developers asked for automated analysis to see trends of usage of an API based on sentiment analysis. We have developed techniques to create time series of positivity and negativity towards an API by analyzing the sentiments expressed about the API in the different forum posts.
3) **API Comparator.** The most-asked tool was a dashboard to compare APIs and to see competing APIs given an API. We developed two techniques to facilitate comparison between APIs. a) We rank APIs by aspect to offer recommendation, such as based on the aspect performance the API Jackson is the most popular for JSON-based tasks in Java, but for usability aspect it is the Google GSON API. b) We further apply collocation algorithm on the forum posts for each API mention and show which other APIs were positively or negative reviewed in the same forum post. This analysis can reveal other similar APIs to developers if they are not satisfied with their initially selected API.
4) **Summarize opinions by topics and paragraphs.** In our pilot survey, we asked developers of their preference to summarize opinions by topics and in short paragraphs. The two options were ranked lower than the aspect-based categorization, but were still considered as useful. We investigated the following existing algorithms to summarize opinions along the two options. Each algorithm takes as input all the opinionated sentences (one bucket for positive and another for negative sentences). a) We applied the widely used topic modeling algorithm, LDA (Latent Dirichlet Allocation) [70] to find topics in the input and to cluster the opinionated sentences by topics. b) We applied four algorithms based on Extractive and Abstractive summarization of natural language texts to produce a summary of the input as a paragraph.

The detail of the summarization algorithms and their evaluation

TABLE 15: Highlights of Findings from Primary Survey RQ1 Closed Questions by Programming Languages

No	Question
1	Do you visit developer forums to seek info about APIs? Research Engineer *1) R:* ***Yes 86.4%****, 2) Java: No 71.4%, 3) Python:* ***Yes 83.3%**** *4) Javascript:* ***Yes 81.2%**** *5) C:* ***Yes 75%**** *6) C++:* ***Yes 83.3%**** *7) C#:* ***Yes 76.5%**** *8) Objective-C:* ***Yes 70.0%**** *9) Ruby:* ***Yes 100%****
3	Do you value the opinion of other developers in the developer forums when deciding on what API to use? *1) R:* ***Yes 78.9%**** *2) Java:* ***Yes 100%**** *3) Python:* ***Yes 83.3%**** *4) Javascript:* ***Yes 84.6%**** *5) C:* ***Yes 87.5%**** *6) C++:* ***Yes 100.0%**** *7) C#:* ***Yes 92.3%**** *8) Objective-C:* ***Yes 100%**** *9) Ruby:* ***Yes 100%****
16	Where do you seek help/opinions about APIs?

	Internal Mailing list	IRC chats	Co-workers	Stack Overflow	Search/Diverse Sources
R	26.3%	31.6%	63.2%	***78.9%****	36.8%
Java	40.0%	20.0%	80.0%	***100.0%****	60.0%
Python	16.7%	16.7%	66.7%	***83.3%****	50.0%
Javascript	30.8%	30.8%	53.8%	***84.6%****	30.8%
C	25.0%	18.8%	56.2%	***87.5%****	43.8%
C++	20.0%	20.0%	60.0%	***100.0%****	50.0%
C#	30.8%	7.7%	***92.3%****	***92.3%****	30.8%
Objective-C	71.4%	14.3%	42.9%	***100.0%****	57.1%
Ruby	25.0%	25.0%	***100.0%****	***100.0%****	25.0%

17	How often do you refer to online forums (e.g., Stack Overflow) to get information about APIs?

	Every day	Two or three times a week	Once a week	One a month	Never
R	***31.6%****	15.8%	21.1%	10.5%	21.0%
Java	40.0%	20.0%	0.0%	40.0%	0.0%
Python	16.7%	***66.7%****	0.0%	0.0%	16.6%
Javascript	***30.8%****	***30.8%****	7.7%	15.4%	15.3%
C	25.0%	***31.2%****	18.8%	12.5%	12.5%
C++	30.0%	***40.0%****	10.0%	20.0%	0.0%
C#	38.5%	30.8%	15.4%	7.7%	7.6%
Objective-C	***71.4%****	0.0%	14.3%	14.3%	0.0%
Ruby	0.0%	50.0%	0.0%	50.0%	0.0%

18	When do you seek opinions about APIs? SAC = Selecting amidst choices, DAR = Determining a replacement, ISF = Improving software feature, RAF = Replacing software feature, DAN = Developing a new API, FAB = Fixing a bug, SAV = Selecting API version, VAS = Validating a selection

	SAC	DAR	ISF	RAF	DAN	VAS	FAB	SAV	Other
R	68.4%	***73.7%****	47.4%	31.6%	47.4%	42.1%	52.6%	31.6%	15.8%
Java	***100.0%****	80.0%	60.0%	20.0%	60.0%	60.0%	100.0%	20.0%	0.0%
Python	***66.7%****	66.7%	***66.7%****	33.3%	50.0%	16.7%	***66.7%****	33.3%	16.7%
Javascript	61.5%	***76.9%****	***76.9%****	38.5%	30.8%	15.4%	***76.9%****	15.4%	15.4%
C	***75.0%****	56.2%	56.2%	56.2%	56.2%	18.8%	50.0%	25.0%	12.5%
C++	***100.0%****	***100.0%****	70.0%	70.0%	80.0%	60.0%	40.0%	20.0%	20.0%
C#	***92.3%****	61.5%	69.2%	46.2%	53.8%	53.8%	69.2%	23.1%	7.7%
Objective-C	***85.7%****	71.4%	71.4%	42.9%	28.6%	28.6%	71.4%	42.9%	14.3%
Ruby	***75.0%****	75.0%	50.0%	50.0%	50.0%	0.0%	50.0%	0.0%	25.0%

was the subject of our recently published paper [51].

8.1.3 Needs for API Usage Summarization (R3)

Developers mentioned that opinions about different API elements can help them better understand the benefits and shortcomings of the API. In our primary survey, developers mentioned that they consider the combination of opinions and code examples posted in Stack Overflow as a form a API documentation. Motivated by this finding, we developed a framework to mine usage scenarios about APIs (R1), where each usage scenario of an API consists of a code example and the reactions (i.e., positive and negative opinions) towards the code example. In our survey, developers mentioned several quality criteria for such usage scenarios while developing an API documentation based on the scenarios, such as offering clues to determine situational relevance of the usage, clarity and correctness of the usage, etc. Indeed, summaries extracted from Stack Overflow can be effective about the purpose and use of API elements (classes, methods, etc.) by building opinion summarizer tools that leverage informal documentation. This motivates future extension of the recent research on augmenting insights about APIs from the forums to the formal API documentation [35]. Such tools can be integrated within IDEs to assist developers during their APIs-based tasks.

As a first step towards producing summaries of usage scenarios of an API as a form of API documentation, we formulated the following tool design requirements:

1) **Show situationally relevant usage scenarios together.** We developed an algorithm to find code examples of an API that are *conceptually similar*, i.e., the tasks supported by the code examples may be closely related to each other. One such example would be, while using the Apache HttpClient API, developers first establish a connection to an HTTP server and then they send or receive messages using the connection [110].
2) **Integrate usage scenarios into API documentation.** In our previous study, we found that the official API documentation can be often obsolete, incomplete, or outdated [6]. In our primary survey, developers asked for a unified documentation by combining API usage scenarios from developer forums with the official API documentation. One such integration would be the Live API documentation, as proposed by Subramanian et al. [64], i.e., link code examples from Stack Overflow into the Javadoc of each API class. We developed algorithms to

TABLE 16: Highlights of Findings from Primary Survey RQ2 Closed Questions by Programming Languages

No	Question
8	Do you rely on tools to help you understand opinions about APIs in online forum discussions? *1) R:* ***No 57.9%**** *2) Java:* ***No 100%**** *3) Python:* ***No 100%**** *4) Javascript:* ***No 76.9%**** *5) C:* ***No 75%**** *6) C++:* ***No 90.0%**** *7) C#:* ***No 84.6%**** *8) Objective-C:* ***No 71.4%**** *9) Ruby:* ***No 100%****
9	If you don't use a tool currently to explore the diverse opinions about APIs in developer forums, do you believe there is a need of such a tool to help you find the right viewpoints about an API quickly? *1) R: No 26.3%, 2) Java: don't know 60.0%, 3) Python:* ***don't know 83.3%**** *4) Javascript:* ***don't know 69.2%**** *5) C:* ***No 50%, don't know 50%*** *6) C++:* ***don't know 80.0%**** *7) C#: don't know 53.8%* 8) Objective-C:* ***don't know 57.1%**** *9) Ruby: don't know 50%, No 50%*

15 An opinion is important if it contains discussion about the following API aspects?
Per = Performance, Sec = Security, Use = Usability, Doc = Documentation, Comp = Compatibility, Comm = Community, Leg = Legal, Port = Portability, Sen = Only Sentiment, Feat = General Features

	Per	Sec	Use	Doc	Comp	Comm	Bug	Leg	Por	Sen	Feat
R	68.4%	68.4%	***78.9%****	73.7%	42.1%	52.6%	78.9%	42.1%	36.8%	21.1%	4.8%
Java	***100.0%****	***100.0%****	***100.0%****	80.0%	***100.0%****	60.0%	80.0%	60.0%	60.0%	0.0%	60.0%
Python	66.7%	***83.3%****	***83.3%****	66.7%	66.7%	50.0%	66.7%	50.0%	33.3%	16.7%	16.7%
Javascript	***76.9%****	69.2%	69.2%	69.2%	61.5%	46.2%	53.8%	30.8%	23.1%	7.7%	46.2%
C	56.2%	62.5%	56.2%	***68.8%****	62.5%	56.2%	68.8%	43.8%	50.0%	0.0%	37.5%
C++	80.0%	70.0%	90.0%	***100.0%****	70.0%	60.0%	80.0%	50.0%	70.0%	40.0%	50.0%
C#	84.6%	***92.3%****	***92.3%****	84.6%	61.5%	61.5%	84.6%	38.5%	30.8%	30.8%	76.9%
Objective-C	71.4%	71.4%	***85.7%****	***85.7%****	71.4%	71.4%	85.7%	57.1%	42.9%	14.3%	28.6%
Ruby	***75.0%****	***75.0%****	***75.0%****	50.0%	0.0%	***75.0%****	50.0%	0.0%	0.0%	0.0%	25.0%

19 What tools can better support your understanding of API reviews in developer forums?
OM = Opinion Miner, SA = Sentiment Analyzer, OS = Opinion Summarizer, AC = API Comparator, TA = Trend Analyzer, CA = Competing APIs, OT = Other Tools

	OM	SA	OS	AC	TA	CA	OT
R	31.6%	10.5%	31.6%	36.8%	47.4%	***52.6%****	0.0%
Java	60.0%	40.0%	60.0%	***80.0%****	***80.0%****	***80.0%****	20.0%
Python	***66.7%****	16.7%	16.7%	50.0%	33.3%	50.0%	0.0%
Javascript	61.5%	38.5%	46.2%	53.8%	***69.2%****	***69.2%****	7.7%
C	37.5%	37.7%	18.8%	56.2%	43.8%	***62.5%****	12.5%
C++	70.0%	50.0%	60.0%	***80.0%****	70.0%	***80.0%****	0.0%
C#	61.5%	69.2%	61.5%	69.2%	***76.9%****	61.5%	15.4%
Objective-C	42.9%	57.1%	57.1%	***100.0%****	85.7%	85.7%	14.3%
Ruby	50.0%	0.0%	25.0%	***75.0%***	50.0%	***75.0%***	0.0%

20 How often do you feel overwhelmed due to the abundance of opinions about an API?

	Always	Sometimes	Rarely	Never
R	5.3%	***31.6%***	31.6%	10.5%
Java	20.0%	***60.0%***	20.0%	0.0%
Python	0.0%	***50.0%***	33.3%	16.7%
Javascript	15.3%	***46.2%***	30.8%	7.7%
C	12.5%	12.5%	***56.2%***	18.8%
C++	0.0%	***70.0%****	30.0%	0.0%
C#	7.7%	***53.8%***	30.8%	7.7%
Objective-C	0.0%	***57.1%***	14.3%	28.6%
Ruby	0.0%	***50.0%***	25.0%	25.0%

No	Question
21	Would a summarization of opinions about APIs help you to make a better decision on which one to use? *1) R:* ***I don't know 36.8%*** *2) Java:* ***I don't know 40.0%, Yes 40.0%*** *3) Python:* ***I don't know 50.0%*** *4) Javascript:* ***I don't know 46.2%*** *5) C:* ***No 50.0%**** *6) C++:* ***Yes 70.0%**** *7) C#:* ***I don't know 46.2%*, Yes 46.2%*** *8) Objective-C:* ***Yes 57.1%*** *9) Ruby:* ***I don't know 75%***

22 Opinions about an API need to be summarized because

	Too many posts	Opinion in missed posts	Missed contradictory opinion	Not enough time	Other reason
R	36.8%	57.9%	42.1%	***63.2%****	5.3%
Java	80.0%	***100.0%****	***100.0%****	60.0%	20.0%
Python	50.0%	***66.7%****	50.0%	50.0%	0.0%
Javascript	46.2%	***61.5%****	46.2%	***61.5%****	15.4%
C	25.0%	56.2%	50.0%	56.2%	6.2%
C++	50.0%	60.0%	***70.0%****	***70.0%****	20.0%
C#	53.8%	***61.5%****	***61.5%****	61.5%	15.4%
Objective-C	71.4%	***85.7%****	57.1%	***85.7%****	14.3%
Ruby	25.0%	***50.0%***	***50.0%***	25.0%	0.0%

23 Opinion summarization can improve the following decision making processes
SAC = Selecting amidst choices, DAR = Determining a replacement, ISF = Improving software feature, RAF = Replacing software feature, DAN = Developing a new API, FAB = Fixing a bug, SAV = Selecting API version, VAS = Validating a selection

	SAC	DAR	ISF	RAF	DAN	VAS	FAB	SAV	Other
R	36.8%	***42.1%****	15.8%	26.3%	21.1%	21.1%	21.1%	26.3%	0.0%
Java	***80.0%****	60.0%	0.0%	20.0%	40.0%	60.0%	80.0%	20.0%	20.0%
Python	***66.7%****	***66.7%****	50.0%	50.0%	***66.7%****	50.0%	50.0%	33.3%	0.0%
Javascript	***69.2%****	84.6%	***69.2%****	46.2%	23.1%	46.2%	38.5%	30.8%	7.7%
C	50.0%	***56.2%****	50.0%	31.2%	25.0%	25.0%	31.2%	25.0%	0.0%
C++	***90.0%****	80.0%	40.0%	50.0%	50.0%	80.0%	20.0%	10.0%	20.0%
C#	***84.6%****	61.5%	53.8%	46.2%	53.8%	61.5%	46.2%	30.8%	23.1%
Objective-C	***85.7%****	***85.7%****	42.9%	42.9%	57.1%	57.1%	28.6%	28.6%	14.3%
Ruby	***75.0%***	50.0%	25.0%	25.0%	25.0%	0.0%	25.0%	25.0%	0.0%

TABLE 17: The major findings from the survey results. CID = ID for Core Finding (to be used for reference in the paper)

RQ ID	Theme	Core Findings	CID
1.1	Needs for seeking opinions about APIs	Developers seek opinions to support diverse development needs	1
		Developers consider the combination of code examples and opinions about an API as a source of API documentation	2
		Developers currently use search engines or Stack Overflow search which are not designed to seek and analyze opinions about APIs	3
1.2	Needs for opinion quality assessment	Developers face challenges to determine the quality of provided opinions	4
		Developers wish opinions to be trustworthy, recent and combined with real-world facts	5
2.1	Needs for tool support	Developers wish for diverse tool supports to analyze opinions, e.g., API comparator, trend analyzer, opinion summarizer, etc.	6
2.2	Needs for summarization	Developers are frequently overwhelmed with the huge volume of opinions about APIs in forums	7
		Developers wish to see opinions about APIs by diverse API aspects (e.g., performance)	8
		Developers wish to get quick, but complete and concise insights about API usage by analyzing both the provided code examples and reviews	9

TABLE 18: The actionable findings used to elicit requirements to design an API opinion and usage summarization engine. R = Requirement, QID = Question ID from primary survey (Questions from Pilot survey are prefixed with a P). CID = ID of the Core findings from Table 17

CID	Actionable Findings/Elicited Design Requirements	Tool Feature	QID
	R1. Needs for Mining of Opinions and Usage Scenarios about APIs		
1,3,6	Opinions about an API need to be collected to assist in diverse development needs	Develop techniques to automatically mine opinions about APIs	18
1-3	Code examples with related opinions need to be collected to assist in API documentation and to show real-world usage experience	Develop techniques to automatically mine usage scenarios about APIs	4, 6, 7, 12, 13
	R2. Needs for Summarization of Opinions about APIs		
7,8	Categorize opinions about an API by aspects, e.g., performance	Aspect based summarization of opinions	11,15
6	Show progression of popularity (i.e., trends) about an API	Statistical Summarization	19
6,9	Need a dashboard to explore and compare competing APIs	(1) Rank competing APIs by popularity. (2) Rank competing APIs by aspects (e.g., popularity by performance) (3) Show APIs co-reviewed in the same posts	13,19, 23
7, 8	Show topics discussed in opinions and provide short summaries as paragraphs	Summarization by Topics, etc.	P17
	R3. Needs for Summarization of Usage Scenarios about APIs		
1,9	Show situationally relevant usage examples together	Concept-Based summarization	12,24
1,9	Integrate usage scenarios into API documentation	Type-Based summarization	12
6,7	Provide situationally relevant information to an API usage	(1) Show APIs used together, (2) Show API elements used together	19,7
	R4. Needs for Quality Assurance about API Opinions		
3,4	Provide situational relevance to an opinion by showing the context	Trace opinions to Stack Overflow posts	6,7
3,4	Show most recent opinions before the older opinions	Rank opinions and usage scenarios by recency	7
3,4	Show ratings to a code example to inform developers of its quality	Show star ratings for each code example based on the reactions of developers towards the post containing the example	7
6	Show contrastive viewpoints about API features	Contrastive Viewpoint Summarization	12
	R5. Needs for a portal for APIs		
1-9	Need a dedicated portal for APIs to explore opinions and usage about APIs	A search and opinion summarization engine for APIs to explore and compare APIs	12

produce type-based usage summaries of an API. In a type-based summary of an API, the usage scenarios of each type (e.g., a class) of the API is grouped and further summarized into situationally relevant clusters.

3) **Provide situationally relevant information to an API usage.** A development task often requires the usage of more than one API. While the above two summaries are focused on producing summaries for an API, we still need information on whether and how other APIs are co-used with our API of interest in the usage scenarios. We used collocation algorithms to find APIs that are co-used together, and other API types that are co-used together with an API type of our interest.

The development and evaluation of the usage summarization algorithm is the subject of our recent publication [53].

8.1.4 Needs for Opinion Quality Analysis (R4)

In our surveys, the developers expressed their concerns about the trustworthiness of the provided opinions, in particular those that contain strong bias. Since by definition an "opinion" is a "personal view, belief, judgment, attitude" [111], all opinions are biased. However, developers associate *biased* opinions with noise defining them as the ones not being supported by facts (e.g., links to the documentation, code snippets, appropriate rationale). While the detection of spam (e.g., intentionally biased opinion) is an active research area [77], it was notable that developers are mainly

concerned about the *unintentional* bias that they associate with the user experience. Developers are also concerned about the overall writing quality.

As a first step towards assisting developers to analyze the quality of the provided opinions, we formulated the following design requirements:

1) **Contrastive viewpoint summarization.** We implement the contrastive opinion clustering algorithm proposed by Kim and Zhai [112] to find pairs of opinions that offer opposing views about an API feature.
2) **Ranking by Recency.** Developers in our surveys were explicitly asked about the necessity for finding more recent opinions about APIs. The reason is that API versions evolve quickly and some old features can become obsolete or changed. We rank the opinionated sentences of an API by recency, i.e., the most recent opinion is placed at the top.
3) **Tracing Opinions to Posts.** Developers in our surveys highlight the necessity of investigating the context of a provided opinion, such as the features about which the opinion is provided, the particular configuration parameters used in the usage example, etc. To enable such exploration, we link each mined opinionated sentence about an API to the specific forum post from where the opinion was mined.

A number of additional research avenues can be explored in this direction, such as, the design of a theoretical framework to define the quality of the opinions about APIs, the development of a new breed of recommender systems that can potentially warn users of any potential bias in an opinion in the forums, etc.

8.1.5 Needs for an API Portal (R5)

In our primary survey, developers explicitly asked for a dedicated portal for APIs where they can search and analyze opinions and usage about APIs that are posted in developer forums, such as Stack Overflow (Q12 in primary survey). Based on the themes and developer needs that emerged from the study, we developed a prototype tool, named Opiner. The tool is developed as an opinion search engine where developers can search for an API by its name to explore the positive and negative opinions provided for the API by the developers in the forum posts.

The Opiner's infrastructure supports the implementation and deployment for all the above requirements (R1-R4). Since the online deployment on October 30, 2017, Opiner has been accessed and used by developers from 57 countries from all the continents except Antarctica (as of July 29, 2018 by Google Analytics).

• **Opiner API Review Summarizer.** In Figure 14, we show screenshots of the user interface of Opiner API review summarizer. The UI of Opiner is a search engine, where users can search for APIs by their names to look for the mined and categorized opinions about the API. There are two search options in the front page of Opiner. A user can search for an API by its name using ①. A user can also investigate the APIs by their aspects using ②. Both of the search options provide auto-completion and provide live recommendation while doing the search. When a user searches for an API by name, such as, 'jackson' in ①, the resulting query produces a link to all the mined opinions of the API. When the user clicks the links (i.e., the link with the same name of the API as 'com.fasterxml.jackson'), all of the opinions about the API are shown. The opinions are grouped as positive and negative (see ⑦). The opinions are further categorized into the API aspects by applying the aspect detectors on the detected opinions ⑧. By clicking on each aspect, a user can see the opinions about the API (see ⑨). Each opinion is linked to the corresponding post from where the opinion was mined (using 'details' link in ⑨). When a user searches by an API aspect (e.g., performance as in ②), the user is shown the top ranked APIs for the aspect (e.g., most reviewed, most negatively reviewed in ④). For each API in the Opiner page where the top ranked APIs are shown, we show the most recent three positive and negative opinions about the API ⑤. If the user is interested to further explore the reviews of an API from ⑤, he can click the 'Explore All reviews' which will take him to the page in ⑦. The system architecture of Opiner API review summarizer is the subject of our tool demo paper [23].

• **Opiner API Usage Scenario Summarizer.** In Figure 15, we show screenshots of Opiner usage scenario summarizer. The user can search an API by name to see the different usage summaries of the API ①. The front page also shows the top 10 APIs for which the most number of code examples were found ②. Upon clicking on the search result, the user is navigated to the usage summary page of the API ③, where the summaries are automatically mined and summarized from Stack Overflow. A user can also click on each of the top 10 APIs listed in the front page. An example usage scenario in Opiner is shown in ④. The reactions included in a usage scenario can be simply a "thank you" note (when the code example serves the purpose) or more elaborated (when the code example has certain limitations or specific usage requirements). In Figure 16, we show an overview of the concept-based API usage summary for the API Jackson in Opiner. In Opiner, each concept consists of one or more similar API usage scenarios. Each concept is titled as the title of the most representative usage scenario (discussed below). In ① of Figure 16, we show the most recent three concepts for API Jackson. The concepts are sorted by time of their most representative usage scenario. The most recent concept is placed at the top of all concepts. Upon clicking on a each concept title, the most representative scenario for the concept is shown in ②. Each concept is provided a star rating as the overall sentiments towards all the usage scenarios under the concept (see ②). Other relevant usage scenarios of the concept are grouped under a 'See Also' (see ③). Each usage scenario under the 'See Also' can be further explored (see ④). Each usage scenario is linked to the corresponding post in Stack Overflow where the code example was found (by clicking on the *details* word after the description text of a scenario).

The summaries page of an API in Opiner also contains a 'Search' box, which developers can use to search for review and usage summaries of another API (e.g., a competing API).

8.2 Effectiveness of Opiner

We investigated the effectiveness of Opiner using both empirical evaluation and user studies. We conducted the empirical evaluation to compute the performance of the mining techniques in Opiner (e.g., the precision of the sentiment detection and opinion association algorithms). We observed a reasonable degree of precision in our mining techniques, such as more than 0.73 in our sentiment detection and .90 in our opinionated sentences to API association algorithm. We conducted a total of six user studies to assess the usefulness of the opinion and usage summaries in Opiner. We found that developers were able to select the right API with more accuracy while using Opiner and leveraging opinion summaries in Opiner. Additionally, we found that developers were able to complete their coding tasks with more accuracy, in less time and

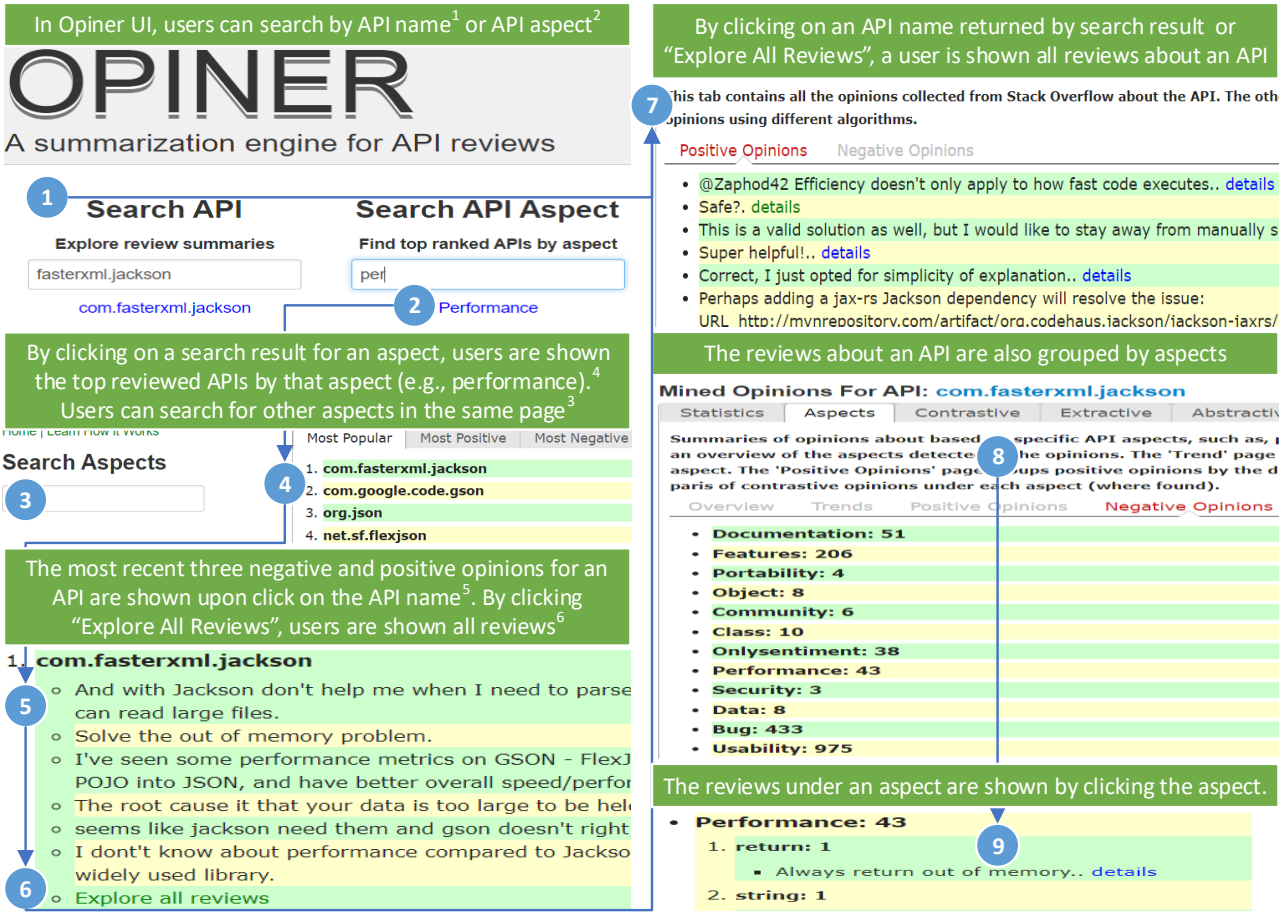

Fig. 14: Screenshots of Opiner opinion search engine for API reviews.

using less effort while using the usage summaries in Opiner. The details of the user studies and empirical evaluation are subject to our recent papers and journals [23], [51], [78], [106], [113]. We summarize the user studies below.

Effectiveness of Opiner Review Summaries. We conducted two user studies to assess the usefulness of the opinion summaries in Opiner and four studies were used to assess the usefulness of the usage scenario summaries in Opiner.

In the first study, we compared our proposed summaries (aspect-based and statistical) against the cross-domain review summarization techniques (summaries in paragraphs, topic-based). We compared the summaries using five development scenarios: 1) Selection of APIs among choices, 2) Documentation of an API, 3) Presentation of an API to others to justify its selection to team members, 4) Staying aware of an API over time, and 5) Authoring a competing API to address the problems of an existing API. A total of 10 professional software developers were asked to rate the usefulness of the summaries for the five tasks. A number of criteria were used to rate, such as usability of the API over other APIs, completeness of the summaries to produce a documentation, etc. The aspect-based summaries were rated as the most useful (more than 85% rating), followed by the statistical summaries (more than 70%). The paragraph-based summaries were considered as the least useful (less than 40% rating).

We conducted the study on-site of a software company. A total of nine software developers from the company participated in the study. The developers were given access to Opiner online tool. The developers were asked to complete the tasks using Opiner and Stack Overflow. Both tasks involved the selection of an API among two competing APIs. For example, the first task asked the developers to pick one of the two APIs (GSON and org.json). The criteria used to select were the usability and licensing restrictions of the two APIs. The developers were asked to consult Stack Overflow and the review summaries in Opiner. The developers made the best decision while using Stack Overflow with Opiner, instead of while using Stack Overflow only. All the developers considered Opiner to be usable and wished to use it in their daily development tasks.

Effectiveness of Opiner Usage Summaries. We conducted four user studies to assess the usefulness of the usage scenario summaries in Opiner. The first study involved the coding of four development tasks and the other involved surveys. A total of 33 professional software developers and students participated in the four user studies (33 in the coding tasks, and 31 of the 34 in each of the surveys).

In the coding study, each participant was given four tasks, for which they wrote code. They used four different development

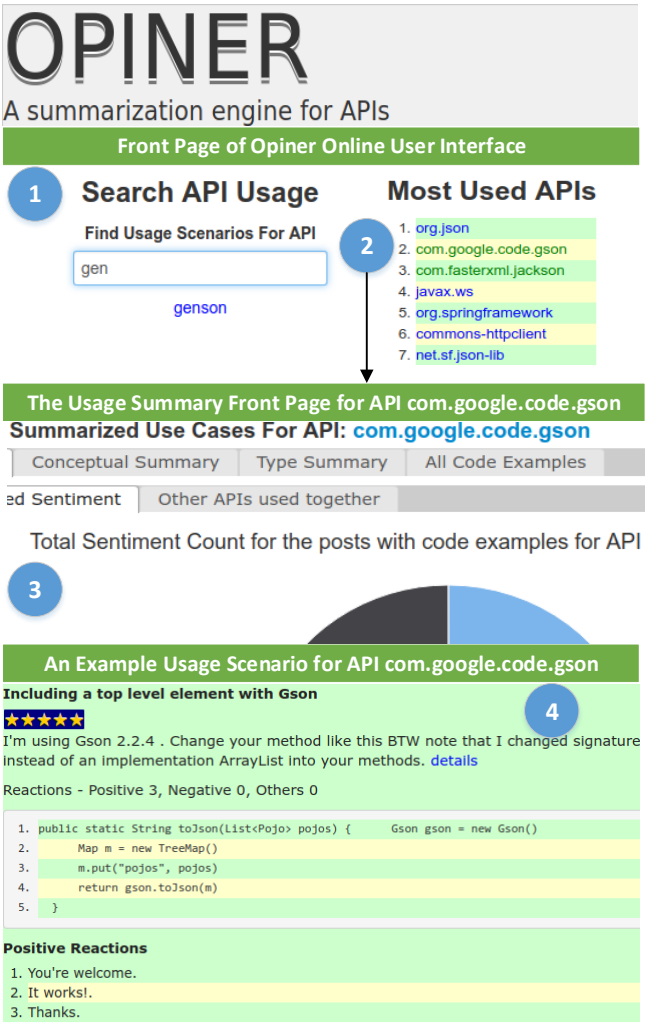

Fig. 15: Screenshots of the Opiner API usage summarizer

resources (one each for the four tasks): Stack Overflow, API official documentation, Opiner usage summaries, and everything including search engines. We observed an average accuracy of 0.62 in the provided solutions while the participants used the Opiner usage summaries. The second best was the everything setting with an accuracy of 0.55, followed by an accuracy of 0.5 when the participants used only Stack Overflow. The participants showed the least accuracy (0.46) while using the API official documentation. In subsequent surveys, more than 80% of the participants agreed that the usage summaries in Opiner can offer improvements over API official documentation and the developer forums. For example, the developers recommended that the usage summaries should be integrated with the API official documentation.

9 Threats to Validity

We now discuss the threats to validity of our study by following the guidelines for empirical studies [114].

9.1 Construct Validity

Construct validity threats concern the relation between theory and observations. In our study, they could be due to the measurement of errors. The accuracy of the open coding of the survey responses

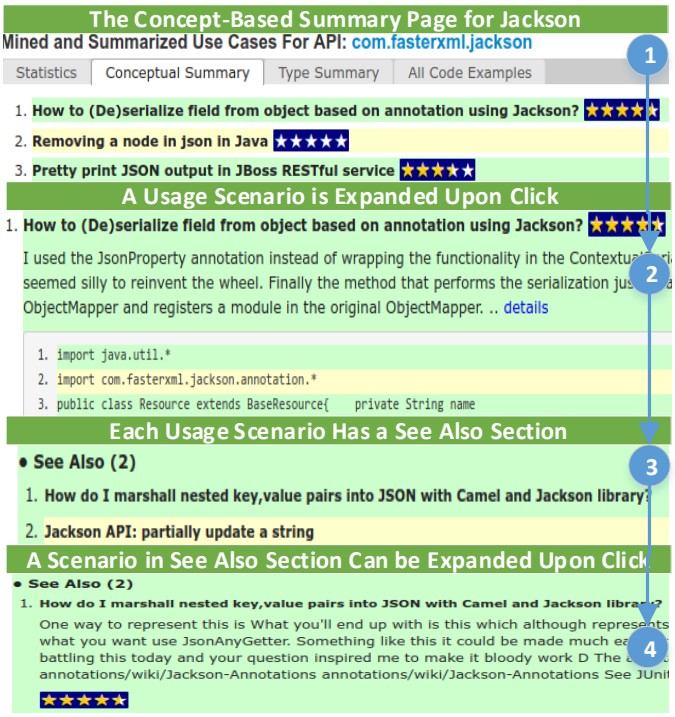

Fig. 16: Concept based summary for API Jackson

is subject to our ability to correctly detect and label the categories. The exploratory nature of such coding may have introduced the *researcher bias*. To mitigate this, we coded 20% of the cards for four questions independently and measured the coder reliability on the next 20% of the cards. We report the measures of agreement in Table 8. While we achieved a high level of agreement, we, nevertheless, share the complete survey responses in an online appendix [89].

Maturation threats concerns the changes in a participant during the study due to the *passage of time*, such as change in development priorities or environments (e.g., moving from open source-based APIs to proprietary APIs). Intuitively, we expect to see greater concentration of opinions about open-source APIs in the forums. None of these concerns are applicable to our surveys, because each survey was supposed to take not more than 30 minutes by each participant.

9.2 Internal Validity

Threats to internal validity refer to how well the research is conducted. In our case, it is about how well the design of the surveys allow us to choose among alternative explanations of the phenomenon. A high internal validity in the design can let us choose one explanation (e.g., tool support to assist in opinion analysis) over another (e.g., no need for a tool) with a high degree of confidence, because it avoids (potential) confounds.

In our primary survey, we sought to avoid confounding factors by asking the participants open-ended questions. The questions with options (i.e., closed questions) were presented to the participants only after they have responded to the open-ended questions. The two types of questions were divided into separate sections, i.e., the participants could not see the closed-ended questions when they answered the open-ended questions. Despite this, we observed similar findings in the responses of the questions. For example, one open-ended question was about the different factors

in an API that can play a role in their decisions to choose the API (Q11). The paired closed-ended question was Q15. In both the responses, we found that developers asked for similar API aspects about which they prefer to seek opinions, such as performance, usability, etc.

Another threat could arise from the placement of the options in a multiple choice question. The threat may influence the participants to agree/disagree with an option more than the other options, such as the option that is placed at the top (i.e., rated first). We did not observe any such pattern in the responses. For example, the first option in our question about tool support in opinion analysis (Q19 in primary survey and Q7 in pilot survey). Both surveys followed the same ordering of the options for the question. Despite this, we observed the lowest rank for the option "sentiment miner" in both surveys (second option) and highest rank for the option "API comparator" (fourth option). In addition, there was no significant difference in the preference of the participants towards a specific tool over another (see Figure 9). Therefore, all such tools were found as favorable by the participants, despite their placements.

9.3 External Validity

Threats to external validity compromise the confidence in stating whether the study results are applicable to other groups.

While our sample size of 900 developers out of a population of 88K developers is small, we received 15.8% response rate in our primary survey. Moreover, the qualitative assessment of the responses offered us interesting insights about the needs and challenges to seek and analyze opinions about APIs. However, it is desirable that future researches replicate our study on a larger and–or different population of developers to make our findings more generic/robust.

Due to the diversity of the domains where APIs can be used and developed, the provided opinions can be contextual. Thus, the generalizability of the findings requires careful assessment. Such diversity may introduce *sampling bias* if developers from a given domain are under- or over-represented. One related threat could arise due to our sampling of developers from GitHub (for pilot survey) and Stack Overflow (for primary survey). The sampling might have missed developers who do not use GitHub or Stack Overflow. As we noted in Section 3.5, we picked developers from GitHub and Stack Overflow because of their popularity among the open-source developers. We observed similar findings between our pilot and primary surveys. Nevertheless, a replication of our study involving developers from other online forums could offer further validation towards the generalization of our findings across the larger domains of software engineering.

Another potential threat of over-representation would be related to all the developers being proficient in only one programming language (e.g., Javascript). This can happen because Javascript has been the most popular language in Stack Overflow over the last six years (according to Stack Overflow survey of 2018 [115]). In our primary survey population of 88,021 Stack Overflow users, we observed that the Stack Overflow posts where the users participated in 2017, corresponded to all the popular languages found in the Stack Overflow surveys. Moreover, while our sample attempted to assign each user to one of the top nine programming languages, those users also participated in the discussions of APIs involving other programming languages. Therefore, their opinions in the primary survey may be representative of the overall Stack Overflow population.

In our primary survey, we only collected responses from developers who visit developer forums to seek information about APIs. As we noted in Section 5, this decision was based on our observation from the pilot survey that developer forums are the primary resource for developers to see such information. In the primary survey, we did not collect additional information from the participants who do not visit developer forums. Further probing of such participants to understand why they do not use developer forums, could have offered us insights into the shortcomings of developer forums to provide such needs. Intuitively, such insights about problems in developer forums can be more concrete from a participant who actually uses developer forums. Therefore, to understand the shortcomings in developer forums, we probed the participants with a number of questions, such as "What are your biggest challenges while seeking opinions about APIs from developer forums?" (Q6), "What areas can be positively/negatively affected by summarization of reviews from developer forums?" (Q13, 14), and "What factors in a forum post can help you determine the quality of the provided opinions?" (Q7). The responses to those questions showed us that developers face numerous challenges while seeking opinions about APIs from forum posts (such as information overload, trustworthiness, etc.). We observed similar findings in our pilot survey. However, we could still have missed important insights from the participants who do not visit developer forums. We leave the analysis of such participants for our future work.

Our survey samples are derived from GitHub and Stack Overflow developers. We picked the pilot survey participants from a list of 4,500 GitHub users. We randomly selected the primary survey participants from a list of more than 88K Stack Overflow users, each of which also had an account in GitHub. We designed the primary survey by leveraging lessons learned from our pilot survey (see Section 4.4). In our primary survey, we attempted to fix each of the problems. For example, we applied *stratification* in our sampling to ensure that we involve developers from Stack Overflow that are *proficient* in different programming languages. The stratification is necessary, because otherwise a random sample from the 88K Stack Overflow users would have picked more developers from a language that is more popular (e.g., Javascript, Java, and Python) among the developers (in terms of the number of users participating in the discussion of the posts related to the language). In addition, we attempted to include the developers who can be considered as *experts* in those programming languages. Intuitively, expertise of a developer for a programming language can be correlated to his reputation in Stack Overflow posts that are related to the language. The higher reputation score a user has, the more likely those reputation scores are provided by many different users in Stack Overflow, and hence the higher likelihood that the user is considered as an expert within the community. An expert user is thus more likely to offer more concrete information about the APIs used in the language, based on real-world experience.

9.4 Reliability Validity

Reliability threats concern the possibility of replicating this study. We attempted to provide all the necessary details to replicate the study. The anonymized survey responses are provided in our online appendix [89]. The complete labels and list of quotes used in the primary survey are also provided in the online appendix.

10 Conclusions and Future Work

Opinions can shape the perception and decisions of developers related to the selection and usage of APIs. The plethora of open-source APIs and the advent of developer forums have influenced developers to publicly discuss their experiences and share opinions about APIs.

To better understand the role of opinions and how developers use them, we conducted a study based on two surveys of a total of 178 software developers. We are aware of no such previous surveys in the field of software engineering. The design of the two surveys and the survey responses form the first major contribution of this paper.

The survey responses offer insights into how and why developers seek opinions, the challenges they face, their assessment of the quality of the available opinions, their needs for opinion summaries, and the desired tool support to navigate through opinion-rich information. We observed the following major findings in our analysis:

1) Developers seek opinions about APIs to support diverse development needs, such as selection of an API among available choices. A primary source of such opinions is the online developer forums, such as Stack Overflow.
2) Developers face several challenges associated with noise, trust, bias, and API complexity when seeking opinions. High-quality opinions are typically viewed as clear, short and to the point, bias free, and supported by facts.
3) Developers feel frustrated with the amount of available API opinions and desire a tool support to efficiently analyze the opinions, such as API comparator, opinion summarizer, API sentiment trend analyzer, etc.

The findings and insights gained from this study helped us to build a prototype tool, named Opiner. The Opiner framework can be used to mine and summarize opinions about APIs in a fully automatic way. We observed promising results of leveraging the Opiner API review summaries to support diverse development needs. The detailed analysis of the survey responses and the findings form the second major contribution of this paper.

Our future work is broadly divided into two directions:

1) **Tool Support by Developer Experience.** As we noted in Section 7.3, less experienced developers show more interest to value the opinions and were also more distinct in their preference of tools to support such analysis (API comparator and Trend Analyzer). We plan to investigate the particular characteristics of the less experienced developers which could motivate them more to value opinions of others and to use the tools. Such insights then can be used to motivate the design of tools and APIs by focusing on the demographic needs. It is thus desirable to replicate our study on a randomized sample of Stack Overflow users, because it may give us higher concentration of less experienced developers than the sample we have used (i.e., based on users with high reputation in Stack Overflow).

2) **Sentiments vs Emotions.** We plan to investigate the role of finer grained emotions (e.g., anger, fear, etc.) in the daily development activities involving APIs. In particular, we are interested in conducting more surveys and developing tools to advance the knowledge on the impact of emotions in API usage analysis.

Acknowledgements

We thank all developers who took part in our survey for their time, participation, and feedback, and Lalit Azad who participated in the open-coding of the primary survey.

References

[1] github.com, https://github.com/, 2013.
[2] M. P. Robillard, "What makes APIs hard to learn? Answers from developers," *IEEE Software*, vol. 26, no. 6, pp. 26–34, 2009.
[3] B. A. Myers and J. Stylos, "Improving api usability," *Communications of the ACM*, vol. 59, no. 4, pp. 62–69, 2016.
[4] J. Stylos and S. Clarke, "Usability implications of requiring parameters in object's constructors," in *Proceedings of the 29th International Conference on Software Engineering*, 2007.
[5] J. Stylos, "Making APIs more usable with improved API designs, documentations and tools," PhD in Computer Sscience, Carnegie Melon University, 2009.
[6] G. Uddin and M. P. Robillard, "How api documentation fails," *IEEE Softawre*, vol. 32, no. 4, pp. 76–83, 2015.
[7] M. P. Robillard and R. DeLine, "A field study of API learning obstacles," *Empirical Software Engineering*, vol. 16, no. 6, pp. 703–732, 2011.
[8] FasterXML, *Jackson*, https://github.com/FasterXML/jackson, 2016.
[9] Google, *Gson*, https://github.com/google/gson, 2016.
[10] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive? empirical study of affectiveness vs. issue fixing time," in *Proceedings of the 12th Working Conference on Mining Software Repositories*, 2015.
[11] M. Måntylå, B. Adams, G. Destefanis, D. Graziotin, and M. Ortu, "Mining valence, arousal, and dominance – possibilities for detecting burnout and productivity?" in *Proceedings of the 13th Working Conference on Mining Software Repositories*, 2016, pp. 247–258.
[12] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *Proceedings of the 22nd International Requirements Engineering Conference*, 2014, pp. 153–162.
[13] E. Guzman and B. Bruegge, "Towards emotional awareness in software development teams," in *Proceedings of the 7th Joint Meeting on Foundations of Software Engineering*, 2013, pp. 671–674.
[14] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014.
[15] N. Novielli, F. Calefato, and F. Lanubile, "The challenges of sentiment detection in the social programmer ecosystem," in *Proceedings of the 7th International Workshop on Social Software Engineering*, 2015, pp. 33–40.
[16] T. Zimmermann, P. Weiβgerber, S. Diehl, and A. Zeller, "Mining version histories to guide software changes," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 429–445, 2005.
[17] A. Michail, "Data mining library reuse patterns using generalized association rules," in *Proceedings of the 22nd IEEE/ACM International Conference on Software Engineering*, 2000, p. 167–176.
[18] B. Dagenais and M. P. Robillard, "Recovering traceability links between an API and its learning resources," in *Proc. 34th IEEE/ACM Intl. Conf. on Software Engineering*, 2012, pp. 45–57.
[19] P. C. Rigby and M. P. Robillard, "Dicovering essential code elements in informal documentation," in *Proc. 35th IEEE/ACM International Conference on Software Engineering*, 2013, pp. 832–841.
[20] A. Bacchelli, M. Lanza, and R. Robbes, "Linking e-mails and source code artifacts," in *32nd International Conference on Software Engineering*, 2010, pp. 375–384.
[21] R. Holmes, R. J. Walker, and G. C. Murphy, "Approximate structural context matching: An approach to recommend relevant examples," *IEEE Trans. Soft. Eng.*, vol. 32, no. 12, 2006.
[22] E. Duala-Ekoko and M. P. Robillard, "Using structure-based recommendations to facilitate discoverability in APIs," in *Proc. 25th Euro Conf. Object-Oriented Prog.*, 2011, pp. 79–104.
[23] G. Uddin and F. Khomh, "Opiner: A search and summarization engine for API reviews," in *Proc. 32nd IEEE/ACM International Conference on Automated Software Engineering*, 2017, pp. 978–983.
[24] J. M. Carroll, P. L. Smith-Kerker, J. R. Ford, and S. A. Mazur-Rimetz, "The minimal manual," *Journal of Human-Computer Interaction*, vol. 3, no. 2, pp. 123–153, 1987.
[25] F. Calefato, F. Lanubile, M. C. Marasciulo, and N. Novielli, "Mining successful answers in stack overflow," in *In Proceedings of the 12th Working Conference on Mining Software Repositories*, 2014, p. 4.

[26] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *In Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 112–121.

[27] S. Lal, D. Correa, and A. Sureka, "Miqs: Characterization and prediction of migrated questions on stackexchange," in *In Proceedings of the 21st Asia-Pacific Software Engineering Conference*, 2014, p. 9.

[28] D. Correa and A. Sureka, "Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow," in *In Proceedings of the 23rd international conference on World wide web*, 2014, pp. 631–642.

[29] B. Vasilescu, A. Serebrenik, P. Devanbu, and V. Filkov, "How social q&a sites are changing knowledge sharing in open source software communities," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, 2014, pp. 342–354.

[30] D. Kavaler, D. Posnett, C. Gibler, H. Chen, P. Devanbu, and V. Filkov, "Using and asking: Apis used in the android market and asked about in stackoverflow," in *In Proceedings of the INTERNATIONAL CONFERENCE ON SOCIAL INFORMATICS*, 2013, pp. 405–418.

[31] B. Bazelli, A. Hindle, and E. Stroulia, "On the personality traits of stackoverflow users," in *In Proceedings of the 29th IEEE International Conference on Software Maintenance (ICSM)*, 2013, pp. 460–463.

[32] A. L. Ginsca and A. Popescu, "User profiling for answer quality assessment in q&a communities," in *In Proceedings of the 2013 workshop on Data-driven user behavioral modelling and mining from social media*, 2013, pp. 25–28.

[33] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Steering user behavior with badges," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 95–106.

[34] Y. Zhang and D. Hou, "Extracting problematic API features from forum discussions," in *21st International Conference on Program Comprehension*, 2013, pp. 142–151.

[35] C. Treude and M. P. Robillard, "Augmenting api documentation with insights from stack overflow," in *Proc. IEEE 38th International Conference on Software Engineering*, 2016, pp. 392–402.

[36] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, pp. 1–31, 2012.

[37] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, p. 33, 2015.

[38] W. Wang and M. W. Godfrey, "Detecting api usage obstacles: A study of ios and android developer questions," in *Proc. 10th Working Conference on Mining Software Repositories*, 2013, pp. 61–64.

[39] G. M. Sarah Rastkar, Gail C. Murphy, "Automatic summarization of bug reports," *IEEE Trans. Software Eng*, vol. 40, no. 4, pp. 366–380, 2014.

[40] G. C. Murphy, M. Kersten, and L. Findlater, "How are java software developers using the eclipse ide?" *IEEE Softawre*, vol. 23, no. 4, pp. 76–83.

[41] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions." *ACM Trans. Softw. Eng. Methodol*, vol. 20, no. 3, pp. 952–970.

[42] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker, "Towards automatically generating summary comments for Java methods," in *Proc. 25th IEEE/ACM international conference on Automated software engineering*, 2010, pp. 43–52.

[43] L. Moreno, J. Aponte, G. Sridhara, M. A., L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for java classes," in *Proceedings of the 21st IEEE International Conference on Program Comprehension (ICPC'13)*, 2013, pp. 23–32.

[44] P. W. McBurney and C. McMillan, "Automatic documentation generation via source code summarization of method context," in *Proceedings of the 21st IEEE International Conference on Program Comprehension*, 2014, pp. 279–290.

[45] L. Guerrouj, D. Bourque, and P. C. Rigby, "Leveraging informal documentation to summarize classes and methods in context," in *Proceedings of the 37th International Conference on Software Engineering (ICSE) - Volume 2*, 2015, pp. 639–642.

[46] G. Murphy, *Lightweight Structural Summarization as an Aid to Software Evolution*. University of Washington.

[47] H. D. Kim, K. Ganesan, P. Sondhi, and C. Zhai, "Comprehensive review of opinion summarization," University of Illinois at Urbana-Champaign, Tech. Rep., 2011.

[48] E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *In Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*, 2013, pp. 562–567.

[49] S. Chang and A. Pal, "Routing questions for collaborative answering in community question answering," in *In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining ACM*, 2013, pp. 494–501.

[50] G. Uddin and F. Khomh, "Automatic mining of opinions expressed about apis in stack overflow," in *IEEE Transactions of Software Engineering*, 2018, p. 35.

[51] ——, "Automatic summarization of API reviews," in *Proc. 32nd IEEE/ACM International Conference on Automated Software Engineering*, 2017, pp. 159–170.

[52] G. Uddin, F. Khomh, and C. K. Roy, "Automatic mining of api usage scenarios from online crowd-sourced forum discussions," in *Technical Report*, 2018, p. 11.

[53] ——, "Automatic summarization of crowd-sourced api usage scenarios," in *IEEE Transactions of Software Engineering*, 2018, p. 25.

[54] S. M. Sohan, F. Maurer, C. Anslow, and M. P. Robillard, "A study of the effectiveness of usage examples in rest api documentation," in *Proc. IEEE Symposium on Visual Languages and Human-Centric Computing*, 2017, pp. 53–61.

[55] O. Barzillay, C. Treude, and A. Zagalsky, *Facilitating Crowd Sourced Software Engineering via Stack Overflow*. Springer New York, 2013, ch. 15, pp. 289–308.

[56] E. Duala-Ekoko and M. P. Robillard, "Asking and answering questions about unfamiliar APIs: An exploratory study," in *Proc. 34th IEEE/ACM International Conference on Software Engineering*, 2012, pp. 266–276.

[57] C. Treude, C. Treude, and C. Treude, "How do programmers ask and answer questions on the web?" in *Proc. 33rd International Conference on Software Engineering*, 2011, pp. 804–807.

[58] F. Shull, F. Lanubile, and V. R. Basili, "Investigating reading techniques for object-oriented framework learning," *IEEE Transactions on Software Engineering*, vol. 26, no. 11, pp. 1101–1118, 2000.

[59] I. Cai, "Framework documentation: How to document object-oriented frameworks. an empirical study," PhD in Computer Sscience, University of Illinois at Urbana-Champaign, 2000.

[60] M. B. Rosson, J. M. Carrol, and R. K. Bellamy, "Smalltalk scaffolding: a case study of minimalist instruction," in *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems*, 1990, pp. 423–430.

[61] H. V. D. Maij, "A critical assessment of the minimalist approach to documentation," in *Proc. 10th ACM SIGDOC International Conference on Systems Documentation*, 1992, pp. 7–17.

[62] Y. Tian, P. S. Kochhar, and D. Lo, "An exploratory study of functionality and learning resources of web apis on programmableweb," in *Proc. 21st International Conference on Evaluation and Assessment in Software Engineering*, 2017, pp. 202–207.

[63] B. Dagenais and M. P. Robillard, "Creating and evolving developer documentation: Understanding the decisions of open source contributors," in *Proc. 18th Intl. Symp. Foundations of Soft. Eng.*, pp. 127–136.

[64] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live api documentation," in *Proc. 36th International Conference on Software Engineering*, 2014, p. 10.

[65] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from stack overflow," in *Proc. 38th International Conference on Software Engineering*, 2016, pp. 392–403.

[66] M. P. Robillard, A. Marcus, C. Treude, G. Bavota, O. Chaparro, N. Ernst, M. A. Gerosa, M. Godfrey, M. Lanza, M. Linares-Vasquez, G. C. Murphy, L. M. D. Shepherd, and E. Wong, "On-demand developer documentation," in *Proc. 33rd IEEE International Conference on Software Maintenance and Evolution New Idea and Emerging Results*, 2017, p. 5.

[67] A. B. Warriner, V. Kuperman, and M. Brysbaert, "Norms of valence, arousal, and dominance for 13,915 english lemmas," *Behavior Research Methods*, vol. 45, no. 4, pp. 1191–1207, 2013.

[68] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 348–351.

[69] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in github: an empirical study," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 352–355.

[70] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[71] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and dynamics of api discussions on stack overflow." Technical Report GIT-CS-12-05, Georgia Tech, Tech. Rep., 2012.

[72] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "How do api changes trigger stack overflow discussions? a study on the android sdk," in *Proceedings of the 22Nd International Conference on Program Comprehension*, ser. ICPC 2014. New York, NY, USA: ACM, 2014, pp. 83–94.
[73] L. Guerrouj, S. Azad, and P. C. Rigby, "The influence of app churn on app success and stackoverflow discussions," in *Proceedings of the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 321–330.
[74] A. M. S. Haiduc, J. Aponte, "Supporting program comprehension with source code summarization," in *In Proceedings of the 32nd International Conference on Software Engineering*, 2010, pp. 223–226.
[75] A. T. T. Ying and M. P. Robillard, "Selection and presentation practices for code example summarization," in *Proc. 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 460–471.
[76] P. Rodeghero, C. McMillan, C. McMillan, N. Bosch, and S. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," in *Proc. 36th International Conference on Software Engineering*, 2014, pp. 390–401.
[77] B. Liu, *Sentiment Analysis and Subjectivity*, 2nd ed. Boca Raton, FL: CRC Press, Taylor and Francis Group, 2010.
[78] G. Uddin and F. Khomh, "Automatic opinion mining from API reviews from stack overflow," *IEEE Transactions on Software Engineering*, p. 35, 2018, under review (Major Revision).
[79] S. Overflow, http://stackoverflow.com/questions/338586/, 2010.
[80] Ohloh.net, www.ohloh.net, 2013.
[81] Sonatype, *The Maven Central Repository*, http://central.sonatype.org/, 22 Sep 2014 (last accessed).
[82] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 168–177.
[83] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. A. Reis, and J. Reyner, "Building a sentiment summarizer for local search reviews," in *WWW Workshop on NLP in the Information Explosion Era*, 2008, p. 10.
[84] C. Treude, F. F. Filho, and U. Kulesza, "Summarizing and measuring development activity," in *Proc. 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 625–636.
[85] M. Miles and A. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE Publications, 1994.
[86] S. Fincher and J. Tenenberg, "Making sense of card sorting data," *Journal of Expert Systems*, vol. 22, no. 3, pp. 89–93, 2005.
[87] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.
[88] D. Freelon, "ReCal2: Reliability for 2 coders," http://dfreelon.org/utils/recalfront/recal2/, 2016.
[89] G. Uddin, O. Baysal, L. Guerrouj, and F. Khomh, *Understanding How and Why Developers Seek and Analyze API-related Opinions*, https://github.com/giasuddin/SurveyAPIReview, 1 December 2017 (last accessed).
[90] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Soft. Eng.*, vol. 30, no. 9, pp. 721–734, 2002.
[91] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *Proc. 2013 International Conference on Social Computing*, 2013, pp. 188–195.
[92] B. Vasilescu, https://meta.stackexchange.com/questions/135104/how-does-so-calculate-email-hash, 2012.
[93] C. T. Commission, *Canadas Anti-Spam Legislation*, http://crtc.gc.ca/eng/internet/anti.htm, 2017.
[94] E. Smith, R. Loftin, E. Murphy-Hill, C. Bird, and T. Zimmermann, "Improving developer participation rates in surveys," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE, 2013.
[97] J. M. James and R. Bolstein, "Large monetary incentives and their effect on mail survey response rates," *Public Opinion Quarterly*, vol. 56, 12 1992.
[95] R. Nisbett and T. Wilson, "The halo effect: Evidence for unconscious alteration of judgments," *Journal of Personality and Social Psychology*, vol. 35, pp. 250–256, 4 1977.
[96] P. Edwards, I. Roberts, M. Clarke, C. DiGuiseppi, S. Pratap, R. Wentz, and I. Kwan, "Increasing response rates to postal questionnaires: systematic review," *BMJ*, vol. 324, no. 1183, p. 9, 5 2002.
[98] W. A. Scott, "Reliability of content analysis: The case of nominal scale coding," *Public Opinion Quarterly*, vol. 19, no. 3, pp. 321–325, 1955.
[99] K. Krippendorff, "Reliability in content analysis: Some common misconceptions and recommendations," *Human Communication Research*, vol. 30, pp. 411–433, 2004.
[100] M. Joyce, *Picking the best intercoder reliability statistic for your digital activism content analysis*, http://digital-activism.org/2013/05/picking-the-best-intercoder-reliability-statistic-for-your-digital-activism-content-analysis/, 2013.
[101] A. J. Viera and J. M. Garrett, "Understanding interobserver agreement: The kappa statistic," *Family medicine*, vol. 37, no. 4, pp. 360–363, 2005.
[102] A. Sharma, Y. Tian, A. Sulistya, D. Lo, and A. F. Yamashita, "Harnessing twitter to support serendipitous learning of developers," in *Proc. IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*, 2017, p. 5.
[103] O. Kononenko, O. Baysal, and M. W. Godfrey, "Code review quality: How developers see it," in *Proc. 38th International Conference on Software Engineering*, 2016, pp. 1028–1038.
[104] C. Bird and T. Zimmermann, "Assessing the value of branches with what-if analysis," in *Proc. ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, 2012, p. Article 45.
[105] S. Overflow, *Sunsetting Documentation*, https://meta.stackoverflow.com/questions/354217/sunsetting-documentation, 2017.
[106] G. Uddin, F. Khomh, and C. K. Roy, "Automatic mining of api usage scenarios from online crowd-sourced forum discussions," in *Technical Report*, 2018, p. 11.
[107] K. Lerman, S. Blair-Goldensohn, and R. McDonald, "Sentiment summarization: Evaluating and learning user preferences," in *12th Conference of the European Chapter of the Association for Computational Linguistics*, 2009, pp. 514–522.
[108] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, "Duplicate bug report detection with a combination of information retrieval and topic modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, 2012, pp. 70–79.
[109] H. U. Asuncion, A. U. Asuncion, and R. N. Tylor, "Software traceability with topic modeling," in *Proc. 32nd Intl. Conf. Software Engineering*, 2010, pp. 95–104.
[110] G. Uddin, B. Dagenais, and M. P. Robillard, "Temporal analysis of API usage concepts," in *Proc. 34th IEEE/ACM Intl. Conf. on Software Engineering*, 2012, pp. 804–814.
[111] Dictionary.com, "Opinion," http://www.dictionary.com/browse/opinion, 2016.
[112] H. D. Kim and C. Zhai, "Generating comparative summaries of contradictory opinions in text," in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 385–394.
[113] G. Uddin, F. Khomh, and C. K. Roy, "Automatic summarization of crowd-sourced API usage scenarios," *IEEE Trans. Soft. Eng.*, p. 25, 2018, under Review.
[114] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering: an introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.
[115] S. Overflow, *Developer Survey Results*, https://insights.stackoverflow.com/survey/2018, 2018.

APPENDIX A
OPEN CODING OF FINAL SURVEY RESPONSES

Table 19 shows the distribution of the categories by the questions.

TABLE 19: The codes emerged during the open-coding of the responses (#TC = Total codes).

	Q4		Q5		Q6		Q7		Q11		Q12		Q13		Q14		Q24		
	#C	#R	#C	#R	#C	#R	#C	#R	#C	#R	#C	#R	#C	#R	#C	#R	#C	#R	#TC
Documentation	11	11	1	1	4	4	54	49	47	43	11	9	6	5	1	1			135
Search			94	79	15	11					12	12	1	1					122
Usability	1	1			2	2			87	56	2	2	2	2					94
Community	11	9			9	7	12	10	31	22	6	3	2	2			2	1	71
Trustworthiness	13	11	1		21	20	8	5	2	1	2	2			21	17	4	3	68
Expertise	31	31	2	2	3	3	13	12	3	3	3	3	4	4			2	2	59
API Selection	13	12			6	6	1	1					30	25	6	6			56
API Usage	17	16			8	8	2	2	10	9	7	6	7	7	2	2			53
Situational Relevance					26	22	12	9	8	7	3	3	3	3			2	2	52
Reputation			1	1	1	1	35	33	5	4									42
Performance	8	8							30	23			3	3					41
Recency			1	1	13	13	6	6			1	1	3	3	3	3			27
Productivity	9	8			1	1							12	11					22
Compatibility					4	4			18	14									22
Aggregation Portal											18	15	1	1					19
Missing Nuances															18	12			18
Legal									15	14									15
Sentiment Statistics			4	4							8	5							12
General Insight					1	1			4	4	1	1	2	2	4	4			12
API Maturity									6	5	3	2							9
API Adoption							2	1	4	4	3	3							9
Portability									5	5			4	4					9
Opinion Categorization											8	8	1	1					9
Opinion Reasoning													1	1	7	6			8
Contrastive Summary											7	7							7
Info Overload					3	3					1	1			2	2			6
Similarity			3	3							2	2							5
Security									2	2	1	1							3
Learning Barrier															3	3			3
Bug									1	1			1	1					2
Extractive Summary											2	2							2
API Improvement													1	1	1	1			2
FAQ											1	1							1
Problem Summaries											1	1							1
Machine Learning											1	1							1
Wrong Info															1	1			1
Lack of Info					1	1													1
Not Sure											10	9	23	22	26	25	1	1	59
Irrelevant	7	4	5	5	10	8	7	4	11	7	25	18	8	7	14	13	6	3	87
Total	121	83	112	83	128	83	152	83	289	83	139	83	115	83	109	83	17	10	1019
# Categories	9		8		17		10		17		21		19		14		5		37

Notes: #C = the number of code, #R = the number of respondents, Q4 = Reasons for referring to opinions of other developers about APIs? Q5 = How do you seek information about APIs in a developer forum?, Q6 = Challenges when seeking for opinions about an API? Q7 = Factors in a forum post to determine the the quality of a provided opinion about an API?, Q11 = Factors of an API that play a role in your decision to choose an API? Q12 = Different ways opinions about APIs can be summarized from developer forums? Q13=Areas positively affected by the summarization of reviews about APIs from developer forums? Q14=Areas negatively affected by the summarization of reviews about APIs from developer forums? Q24= Reasons to not value the opinion of other devs.