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

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 developer 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 a survey of 64 professional developers. The results provide 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, as well as their desired tool support to navigate through opinion-rich information.

Index Terms—Opinion mining; API informal documentation; opinion summaries; survey; opinion quality; developer 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 38 million public repositories. We can observe a radical increase from the 2.2 million active repositories hosted in GitHub in 2014. Developers share their projects and APIs via other portals as well (e.g., Ohloh [2], Sourceforge [3], Freecode [4]). For example, Ohloh currently hosts more than 650,000 projects, demonstrating a 100,000 increase from 2015. We used the GitHub and Ohloh APIs to generate the statistics.

With a myriad of APIs being available, developers now face a new challenge — how to choose the right API. To overcome this challenge, many developers seek help and insights from other developers. Fig. 1 presents the screenshot of seven Stack Overflow posts (four answers, three comments). The oldest post (at the top) is dated November 06, 2009 and the most recent one (at the bottom) is February 10, 2016. These posts express developers' opinions about two Java APIs (Jackson [5] and Gson [6]) offering JSON parsing features for Java. None of the posts contain any code snippets. The first answer (A1) representing a positive opinion about the Gson API motivates the developer 'binaryrespawn' to use it (C1). In the next answer (A2), the user 'StaxMan' compares Gson with Jackson, favouring Jackson for offering better support, and based on this feedback, 'mickthomson' (A3) decides to use Jackson instead of Gson. Most of the posts (A2–A4) 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 (C3). This example illustrates how developers share their experiences and insights, as well as how they influence and are influenced by the 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 her 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

Fig. 1: Example of a Stack Overflow discussion about two APIs.

analysis [7], determining developers burnout [8], improving software applications [9], and developing awareness tools for software development teams [10], [11], [12] (Section 5). Research on APIs has produced significant contributions, such as automatic usage inference mining [13], [14], automatic traceability recovery between API elements (types, methods) and learning resources [15], [16], [17], and recommendation systems to facilitate code reuse [18],

TABLE 1: Sentiment coverage in Stack Overflow posts tagged as "Java" and "Json".

Posts	Opinionated	With Code	Opinionated No Code	Sentences	Positive	Negative	Words	Sentiment words
22,733	15,206	5,918	10,424	87,033	15,845	11,761	10,77,033	41,959

[19] (refer to Section 5).

However, there is very little research that focuses on the analysis of developers' perception about API opinions and how such opinions affect their decisions related to the APIs. As illustrated in 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. In fact, we observed that more than 66% of Stack Overflow posts that are tagged "Java" and "Json" contain at least one positive or negative sentiment (refer to Table 1). We also observed that most of these posts do not contain any code examples. We compiled the statistics based on the Stack Overflow data dump of January 2014 [20] and by only parsing the posts that have positive scores. We identified the sentiment words by matching more than 6,000 sentiment lexicons developed by Liu et al. [21] against the words in the posts. Indeed, Bacchelli et al. [22] also observed that the textual contents of software artifacts are useful sources for gaining important insights about the software. With the growing popularity of developer forums, such as Stack Overflow, the number of posts discussing APIs would also increase. For example, between January 2014 and August 2016, the number of questions tagged as "Java" in Stack Overflow has increased from half a million to more than one million.

Given the significant presence of sentiments in the forum posts, we conducted a study based on a survey involving professional software developers with the goal of understanding what developers think about opinions of others and whether they find such information useful. Such developers' insights can contribute to the development of an empirical body of knowledge on the topic and to the design of tools that analyze opinion-rich information. Through a thorough exploratory analysis of the survey responses, we address the following research questions:

- RQ1: *How do developers seek opinions about APIs?*
  We refine this question into the following subquestions:
  - RQ1.1: *Where do developers seek opinions?*
  - RQ1.2: *What motivates developers to seek opinions?*
  - RQ1.3: *What challenges do developers face while seeking opinions?*
- RQ2: *What are the developers' needs for summarization of opinion-rich information?*
  We formulate the following subquestions to answer this:
  - RQ2.1: *What problems in opinion-rich resources motivate the need for opinion summarization?*
  - RQ2.2: *How summarization of API opinions can support developer decision making?*
  - RQ2.3: *How do developers want opinion summaries to be organized?*
- RQ3: *How do developers assess the quality of the provided opinion?*
- RQ4: *What tool support is needed to help developers with analyzing and assessing API-related opinions?*

This paper makes the following contributions:

1) A detailed analysis of the survey responses that provides insights into:
   - How professional developers seek and analyze opinion-rich API information?
   - Need of professional developers for opinion summarization
   - Assessment of the quality of the available opinions by professional developers
   - Expectations on the necessary tool support to help professional developers navigate through developer forums.
2) The study design involving professional developers and its corresponding dataset of 64 responses are available at http://bit.ly/2bGPz3H.

Paper organization. Section 2 describes the survey design, participants, and data analysis. In Section 3, we present our findings. Section 4 discusses implications of our work and future research directions. Section 5 summarizes relevant related work. Finally, Section 6 concludes our paper.

2 Methodology

To address our research questions, we conducted an exploratory qualitative study that involved data collection through an online survey with professional software developers. This section describes the survey design, the participants, and the analysis preformed to analyze the responses in detail.

2.1 Survey Design

The survey consisted of 24 questions: demographic questions, eight multiple-choice, five mandatory Likert-scale questions, and eight open-ended questions. The most important survey questions are indicated in Table 2.

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.). These questions are also available in our replication package.

Even though, the answers to the open-ended questions were optional, almost all of the participants answered one or more of those questions.

The main goal was to solicit developers' insights on how they consider opinions of others when they evaluate APIs in online development forums (e.g., Stack Overflow). We also wanted to identify and assess their needs for opinion mining and summarization of APIs. The survey consisted of four main parts: three questions about the developers' demographic background and work practices, eight multiple-choice questions related to how and why developers refer to the opinions that relate to the API-related informal documentation; five Likert-scale questions related to different aspects of opinion seeking and summarization of online help, and eight follow-on open-ended questions to allow developers to further elaborate on the issues raised by the multiple choice questions. Participants were asked to spend 5–10 minutes to complete the survey.

TABLE 2: Survey questions (excerpt).

P1	Opinion Needs
1	Do you value opinion of other developer when deciding on what API to use? *(yes/no)*
2	Where do you seek help/opinions about APIs? *(five sources and others)*
3	How often do you refer to online forums (e.g, Stack Overflow) to get information about APIs? *(five options)*
4	When do you seek opinions about APIs? *(5-point likert scale for each option)*
5	What are other reasons of you referring to the opinions about APIs from other developers? *(text box)*
6	What is your biggest challenge while seeking opinions about an API? *(text box)*
P2	**Tool Support**
7	What tools can better support your understanding of opinions about APIs in online forum discussions? *(5-point likert scale for each option)*
8	What other tools can help your understanding of opinions about APIs in online forum discussions? *(text box)*
P3	**Summarization Needs**
9	How often do you feel overwhelmed due to the abundance of opinions about an API? *(four options)*
10	Would summarization of opinions about APIs help you to make a better decision on which one to use? *(yes/no)*
11	Opinions about APIs need to be summarized because? *(5-point likert scale for each option)*
12	Opinion summarization can improve the following decision making processes. *(5-point likert scale for each option)*
13	What other areas can be positively affected having support for opinion summarization? *(text box)*
14	What other areas can be negatively affected having support for opinion summarization? *(text box)*
P4	**Summarization Type**
15	An opinion is important if it contains discussion about one or more of the following API aspects? *(5-point likert scale for each option)*
16	What are the other factors you look for in API opinions? *(text box)*
17	What type of summarization would you find most useful? *(five options)*
18	How many keywords do you find to be sufficient for topic description? *(five options)*
19	What are the other ways opinions about APIs should be summarized?*(text box)*
P5	**Opinion Quality**
20	How do you determine the quality of a provided opinion in a forum post? (e.g., Stack Overflow) *(seven options)*
21	What other factors in a forum post can help you determine the quality of a provided opinion? *(text box)*

2.2 Participants

We targeted the GitHub developer community. We sent survey invitations to 2,500 randomly sampled GitHub users who satisfied the following criteria: (1) Shared or contributed to at least one public repository, and (2) Shared at least one gist (3) Followed by one or more other GitHub user(s). With the first two criteria, we ensured a user who is active in development and who is also active in sharing relevant knowledge through gist with others. We used the last criterion to ensure that the user is known and popular to other developers in GitHub, i.e., it is more likely that other developers consider the contribution of the developer as valuable.

In addition to the email recruitment, we have posted the link to our survey in a post on the Slashdot (March 09, 2016) and Reddit (March 13, 2016) forums. We did not succeed in recruiting developers from the forums as user engagement in such forums is mainly driven by the top voted posts. Since our survey promised no rewards, it fell out of the top pages within a few minutes.

The survey was open for about seven weeks (from March 09 to April 30, 2016). In total, we received 64 responses. Since we posted the link to our survey on two developer forums, it is difficult for us to accurately calculate the response rate. 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 were not interested in participating due to the lack of any incentives. If we consider only our email-based recruitment efforts, the response rate is 2.64%. Previously, a low response rate from the GitHub users was also observed by Treude et al. [23], who achieved a response rate of 7.8% despite incentivizing their survey by offering gift cards. Possible explanation of our lower response rate is also the GitHub users' "sensitivity" towards emails from researchers as had been widely discussed in the research community early this year, just before we sent the survey invitations (refer to: https://github.com/ghtorrent/ghtorrent.org/ issues/32).

The beginning of the survey consisted of background-related questions. By analyzing the responses, we found that we had successfully targeted *active developers with various levels of experience*: 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.

2.3 Survey Data Analysis

To analyze the survey data, we applied an open coding approach. As we analyzed the quotes, themes and categories emerged and evolved during the open coding process [24].

Author U created all of the "cards", splitting the responses for eight open-ended questions into 173 individual quotes; these generally corresponded to individual cohesive statements. In further analysis, authors U and B acted as coders to group cards into themes, merging those into categories. We analyzed each such 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

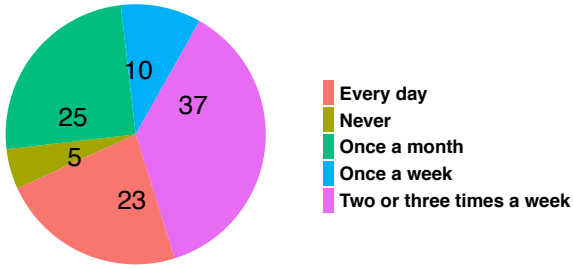

Fig. 2: The frequency (percent) of seeking API opinions.

selected two popular reliability coefficients for nominal data: percent agreement and Cohen's Kappa [25].
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 [26] 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). The average percent agreement values for the applicable survey questions are reported in Table 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.

3 Results

During the open coding process of eight open-ended questions, 29 categories emerged (excluding "irrelevant"). Table 3 reports the number of quotes and respondents, the questions, and the average percent agreement for each question.

3.1 How Developers Seek Opinions about APIs (RQ1)

When we asked developers whether they seek opinions about APIs, 95% of the respondents said that they value opinions of other developers. Next, we identified the key sources of opinion-rich information about APIs.

3.1.1 Sources for opinions

Developers pointed out two prominent sources of such information – developer forums (e.g., Stack Overflow) and co-workers (both representing 83.3% of responses). 33.3% of developers identified internal mailing lists as a major source, while 26.7% listed IRC chats. 35% also described several other sources, such as: (1) trusted users and friends; (2) GitHub; (3) usage stats (e.g., stars) of online API repositories (e.g, ruby-toolbox, npmjs, etc.); (4) meetups; (5) blog postings and white papers; (6) Google; (7) API documentation; and (8) self-help (intuition, trying myself).

Since online developer forums are seen as the primary source of opinion-rich API information, we asked developers how frequently they seek API-related help in the online forums (Figure 2). Most of the developers (36.7%) answered that they look for opinions two or three times a week, 25% once a month, 23.3% every day, and 10% once a week. The 5% of respondents who indicated that they do not find opinions to be useful, all of them mentioned that they use online forums to seek help/opinions about APIs at least once a month.

3.1.2 Factors motivating opinion seeking

We asked developers about the factors that motivate them to seek opinions. We solicited opinions using a five-point Likert-scale question and then probed more in-depth information through an optional follow-up open-ended question. The proposed answers to our Likert-scale questions were compiled from the factors that we identified through our own exploration of opinions in the online forums. The open-ended questions provided developers an opportunity to specify additional factors that were not covered by the Likert-scale question.

The analysis of the Likert-scale question (Figure 3) shows that developers find selection-related factors (i.e., determining replacement of an API and selection of an API among choices) to be most influential for seeking API opinions (92% and 91% of the non-neutral responses, respectively). Developers also seek opinions when they need to improve a software feature, for which they hope to find a well-reviewed API (82%). 78% of the developers agree that they turn to online help when they need to fix a bug. 78% of responders 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 or replacing one version of an API with another version (55% and 34%, respectively).

The analysis of the open-ended question revealed several software engineering aspects that developers believe can be impacted by collecting opinions about APIs. The largest category identified in the responses is *reasoning about API usage*. As explained by R53 *"seeking to find a either way to do something in this API or the 'natural' way to do something in a given API"*. When learning from others, developers want to find opinions about specific *"program flow"* (R25). Getting to know specific edge cases is also important for them. R57 explains: *"you don't tend to figure out the subtle effects of an API until you stumble across some problem yourself"*. When learning about proper usage, developers are interested to know *"how others used the API"* (R45), because *"not all interactions can be obvious"* (R12).

The second major category is related to the API *documentation*, e.g., documentation support and quality. According to R12, *"the biggest area is to deal with bad documentation"*. This theme confirms the findings from the previous studies that developers desire better API documentation [27], [28].

The *API selection* is a recurrent theme in the responses, such as the discovery of competing but previously unknown APIs. To R58, such opinions can also provide insights into the overall trends, *"...are people tending to move away from it? Then it may be better to adopt something else."*. *Saving of time* is the key motivation in such discovery. According to R57, *"it's valuable to do a quick search to get a sense for what is going to be problematic, what will be hard, and what the API is really tuned for"*. Learning about the limitations of an API can also facilitate the API selection. To R62, it is more about how quickly a decision can be made through such discussions, *"I save time on discovering the same conclusions (i.e., that the API isn't performant, or has other limitations)"*.

Developers said that they want to hear opinions that *recommend new features* to the existing APIs, as R23 explains his reason: *"to see if there's sufficient general consensus to recommend changes to the authors"*. *Standardization* of APIs is a big concern to the developers, such as complying to it or promoting it to others, *"to adhere to an industry standard and make it easier to understand for those consuming"* (R12).

TABLE 3: The list of categories that emerged during the open coding of open-ended questions.

Category	Q8		Q9		Q11		Q16		Q17		Q19		Q22		Q24	
	#Qt	#R	#Qt	#R	#Qt	#R	#Qt	#R	#Qt	#R	#Qt	#R	#Qt	#R	#Qt	#R
API usage	5	5	2	2	–	–	3	3	1	1	6	6	3	2	–	–
API experts/expertise levels	–	–	1	1	2	1	1	1	–	–	3	3	1	1	2	2
Documentation	2	2	3	3	3	3	2	2	–	–	–	–	1	1	12	12
Trustworthiness/reputation	–	–	–	–	–	–	3	3	4	3	1	1	–	–	2	2
API maturity/change	–	–	–	–	1	1	2	1	–	–	2	2	1	1	–	–
Opinion reasoning	–	–	–	–	6	5	–	–	– 1	1	–	–	1	1	–	–
Standardization	–	–	1	1	1	1	–	–	–	–	2	2	–	–	–	–
Usability	2	2	1	1	–	–	–	–	–	–	4	4	–	–	–	–
API popularity	–	–	–	–	1	1	–	–	–	–	–	–	1	1	–	–
API selection	3	3	–	–	–	–	1	1	–	–	–	–	–	–	–	–
API suitability	2	2	1	1	–	–	–	–	–	–	–	–	–	–	–	–
Community engagement	1	1	–	–	2	2	–	–	–	–	–	–	–	–	–	–
Information overload	–	–	5	5	–	–	–	–	4	3	–	–	–	–	–	–
Learnability	–	–	–	–	2	2	–	–	–	–	1	1	–	–	–	–
Saving time	2	2	–	–	–	–	1	1	–	–	–	–	–	–	–	–
Situational relevance	–	–	2	2	–	–	–	–	–	–	–	–	–	–	2	2
Testing	–	–	–	–	–	–	–	–	–	–	1	1	1	1	–	–
Adoption risk analysis	–	–	1	1	–	–	–	–	–	–	–	–	–	–	–	–
API categorization	–	–	–	–	1	1	–	–	–	–	–	–	–	–	–	–
API quality performance	1	1	–	–	–	–	–	–	–	–	–	–	–	–	–	–
API version	–	–	–	–	–	–	–	–	–	–	–	–	–	–	1	1
Biased opinion	6	6	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Change recommendation	1	1	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Dependency info	–	–	–	–	4	3	–	–	–	–	–	–	–	–	–	–
Extractive & adaptive summary	–	–	–	–	6	2	–	–	–	–	–	–	–	–	–	–
General insight	2	2	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Limitations	2	2	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Staying up to date	–	–	–	–	2	2	–	–	–	–	–	–	–	–	–	–
Summarize by time	–	–	–	–	1	1	–	–	–	–	–	–	–	–	–	–
Irrelevant	1	1	1	1	1	1	1	1	2	1	1	1	3	2		
Total	32	21	31	28	33	17	14	9	12	9	21	16	12	7	18	15
# Categories	13		10		14		8		5		9		8		5	
Average Percent Agreement	**94.40%**		**98.20%**		**100.00%**		–		–		**92.00%**		–		–	

Notes: #Qt: the number of quotes, #R: the number of respondents, Q8: reasons for seeking opinions, Q9: challenges while seeking opinions, Q11: tool support for opinion analysis, Q16: areas positively affected by summarization, Q17: areas negatively affected by summarization, Q19: APIs aspects sought after in opinions, Q22: ways opinions about APIs can be summarized, Q24: opinion quality

Developers mentioned that gaining insights into the API *quality* attributes is important. To R27, opinions are *"efficient way of gaining an overview of the actual quality of a given API"*. Such attributes can include the *"operational performance"* (R61) and *usability*. According to R13, *"I want to learn about ease of use"*.

Developers desire to learn about *sustainability* of an API from the opinions, e.g., its newness and *maturity*, or *community engagement*. R58 wants to stay aware of the issues related to a new API, *"is it new enough that we're likely to encounter issues?"*. Opinions can be used to learn about the overall support towards an API *adoption*, *"to get an idea of how mature and/or well supported the API will be in the future"* (R58).

Overall, developers feel that they can be more informed of *expert insights* through the opinions from other developers, because, according to R24, *"they have experience"*. Through such shared knowledge, developers hope to *stay up-to-date*.

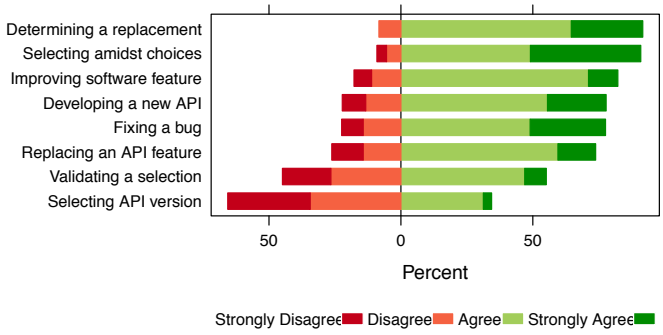

Fig. 3: Developer motivation for seeking opinions about APIs.

3.1.3 Challenges in opinion seeking

While developers find opinions about APIs useful, they also cite various difficulties associated with seeking such opinions. The major challenge is identification of *biased opinion*. As explained by R4, *"judging authoritative opinions, backed by facts, from just preference can be a challenge"*. An opinion can be biased if it comes *"either from the vendor themselves, or by a user of it. Vendors will only give the positive points, and users only have experience with their use case. In both cases, it's hard to see the whole truth."* (R58). Removal of obsolete discussions and promotional material from the posts can help developers trust the opinions more. Similarly, developers discuss challenges related to *trustworthiness* of the opinion provider and sources. Such lack of trust does not necessarily boil down to the assessment of honesty, rather it is associated with the subjective nature of opinions in general. The challenge is to *"evaluate the experience/trustworthiness of the persons giving advice/opinion. People on all public mediums are very opinionated, so evaluating the opinions of others*

is difficult." (R1). Finding suitable means to assess trustworthiness and bias can be challenging, but R29 offers a solution, *"I end up consulting many sources and if the same answers come back over and over, I accept them." Finding an expert* for a given API among many opinions can also be a non-trivial task.

The presence of *noise* in the opinions is another challenge. As APIs evolve, opinions can become obsolete or useless. As explained by R62, the challenge is *"sifting through the many useless posts of OLD posts that may be irrelevant"*. Low quality information can also be present. While searching for facts in opinions, developers also find it difficult to remove the clutter, such as mere emotions. The *lack* of standard guidelines on how APIs can be designed or compared is also cited as a contributing factor when assessing the usefulness of an opinion. R60 proposes on *"having a valid and accepted convention set out by something like IEEE, we read comments from people online on how to design these APIs for things such as the URL path, and there is no real standard"*.

An opinion with a complete and objective picture about an API is typically hard to come across. Thus, *API selection* based on opinions becomes a peril. The selection can become even more challenging due to the difference in use cases. As explained by R13, *"it is common that the specific use case I have in mind for the API is not one that others have already done. Therefore, it becomes difficult trying to determine whether or not the API will be suitable for my project"*. Another challenge is a *cold start problem*, namely, when a developer is completely new to a domain and she does not know what to search through the available tools, *"where to start, I'm sometimes unaware of what is even available"* (R3). While an *educated guess* in selecting an API is seen as a last resort, such a decision is likely a bad choice — *"most of the time we take an educated guess as to our best choice, but more usually there is a single de factor choice for solving a problem"*.

Challenges also arise when a complete picture is not evident due to the *complexity of the API* or misalignment of scope in the provided opinions. Another problem is related to the *situational relevance*, when it's hard for developers to *"determine how close another's person use case and experience mirror my own"* (R10). Environmental *suitability* is a challenge, such as when the opinion is provided for an operating environment that is different from the environment for which the opinion is asked for. As explained by R61, *"environmental parity, i.e., different operational environments can be profound impact on the suitability of particular API"*. The *lack of good documentation* about APIs, a recurring category, can also be challenging when assessing opinions about API usage. Overall, opinions with *real-world examples* are much preferred.

Finally, finding clarity over the *risk of adoption* of an API can be challenging from a subset of all given opinions about the API. According to R46, the problem is how *"to find the right balance between usability and feature-richness"*.

RQ1: Most developers seek opinions on a weekly basis. Online developer forums and co-workers remain the primary sources of opinions about APIs. Trust, bias, noise, and API complexity are common challenges in opinion seeking.

3.2 Developer Needs for API Opinion Summarization (RQ2)

We asked developers whether they feel overwhelmed due to the abundance of opinions about APIs in the forums. Out of 86.7% of the respondents who replied with a "yes" to the question, 5% reported to be overwhelmed every time they look for opinions about an API, and 48.3% reported to feel overwhelmed some time. In our subsequent question, we asked them whether a summarization of the opinions about APIs could alleviate such situation. 83.3% of the respondents agreed that opinion summaries could indeed help them.

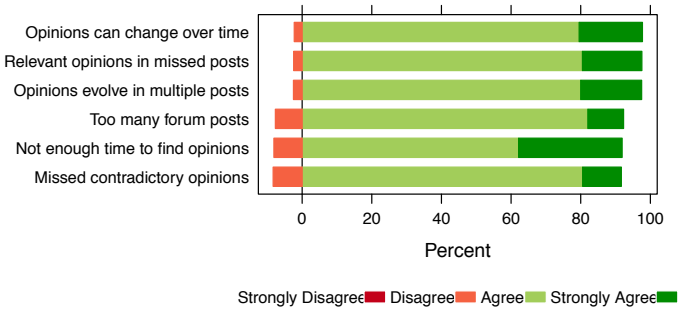

Fig. 4: Developer needs for opinion summarization about APIs.

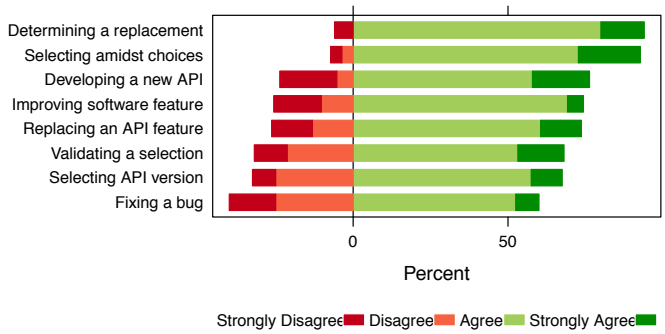

Fig. 5: Impact of opinion summaries on decision making about APIs.

3.2.1 Needs for opinion summarization

We explored the relative benefits and disadvantages of opinion summarization by asking developers why opinions need to be summarized and how these opinions may impact their decision making.

The analysis of the Likert-scale question (Figure 4) shows that nearly all developers agree that opinion summarization is much needed for several reasons. Since opinions change over time or across different posts, developers want to be able to track changes in API opinions and follow how APIs evolve. The vast majority of responders also 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 myriad of opinions (coded as "information overload") expressed via various developer forums. Developers 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.

We next asked developers about how opinion summaries can support their tasks and decision making (Figure 5). More than 90% 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 develop a new API to address the needs that are currently not supported, improve a software feature, replace an API feature, validate choice of API,

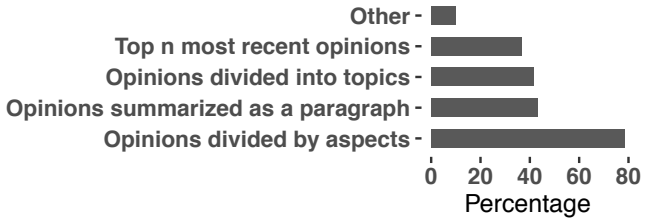

Fig. 6: Opinion summarization types desired by developers.

and select the right version of an API (76%, 74%, 74%, 68% and 68%, respectively). Fixing a bug, while receiving 60% of positive responses, 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 analysis of the open-ended question revealed several other areas that developers believe have a positive impact on opinion summarization. Some developers feel that understanding opinions can help them in selecting the right API or *"identifying the best forum for each API"* (R53). Others believe that opinions are rich in offering various *API usage* examples and describing the *API complexity*. Several developers express their concerns with assessing the *trustworthiness* of a provided opinion and feel that the quantification of opinions can help them in deciding whether *"an advice (positive or negative) has been done by someone really understanding the problem"* (R56). Developers also find that understanding opinions can *"help in the design of [API] replacements/splits"* (R23).

Developers also identify several pitfalls when relying on the opinions. They worry that in some cases opinions *"represent noise on the web"* and that summarization of opinions can *"over-represent certain opinions"* (R57). As R32 mentions, *"summarization is only helpful if it's trusted. How do I know if it's competent, well sourced, with no hidden objectives or sponsorship?"*. Opinion summaries can become overhead as developers would need to consider them during their tasks (e.g., selecting the right API), *"I'd have to adapt to their existence and figure out how to weight them or what things they work for and what they don't"* (R57). Another possible drawback of opinion summaries is seen as making *"API features to be hidden if an API is overly complex"* (R25). Since opinions change over time, developers see the need for opinion versioning to be able to differentiate between old and new ones. This insight, while surprising and interesting, expresses developer needs for a possible opinion version control system.

3.2.2 Summarization types

According to the above mentioned results, developers see several benefits of summarizing opinions about APIs. Next, we ask for the developer feedback on how they want opinion summaries to be organized. Participants were presented with a multiple choice question with a "Select all that apply" option. The results are presented in Figure 6. Majority of the developers (78%) prefer API opinions to be organized according to the API aspects (e.g., performance, usability, etc.). 45% of participants want to see a small paragraph summarizing most positive and negative opinions about a specific API. 43% of developers prefer to model opinion summaries according to the topics discussed (e.g., "simple, binding, gson, mapping, stream"), while 37% would like to have top N opinions to be displayed based on a certain criteria (e.g., most recent ones). Overall, we can see that developers' opinions on how textual summaries can be organized and presented to them vary. Yet, API aspects seem to be the key criterion for modelling opinion summaries.

In the open-ended question, some developers preferred other forms of organizing opinions about APIs. R62 prefers to have opinions prioritized by the expertise level, *"experts should be weighed higher, i.e., based on answer scores for the topic in question"*. Two participants expressed desire for having opinions summarized by a specific feature of the API. Other developers want to organize opinions based on the presence of examples and tutorials, *"simple tutorials with source code would give a chance to evaluate an API"* (R31). API maturity, adoption rate, testing and positive/negative reasoning were among other suggested forms of organizing opinions.

3.2.3 Summarization by API aspects

So far, the analysis of the survey data showed that developers believe that opinion summaries can be useful in supporting their decisions about APIs. We next seek developers' insights into what aspects of an API need to be addressed in the opinion for it to be considered by the developers. Through a mandatory multiple choice question and an optional open-ended question, we asked participants about the content of a useful opinion. The results of the relevant Likert-scale survey question are summarized in Figure 7. 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.

Some of the additional items that developers look for in API opinions are: real-world examples and scenarios of usage, *"reputation of opinion provider"* (R61) and *"experience"* (R23), presence of good tests, code style, API maturity (*"how long it's been active for"* (R25)), *"extensibility of an API"* (R29), API's learning curve, and discussion of strong vs. weak points such as *"good at x, bad at y"* (R29).

RQ2: Developers agree that opinion summaries can support their API-related tasks and decisions, yet a proper quality control (trust, noise) must take place. Summarization by the API aspects is the preferred type of opinion summaries.

3.3 Developer Assessment of API Opinion Quality (RQ3)

We asked developers two questions about how they determine the quality of the provided opinion in a forum post (e.g., Stack Overflow). The first was a multiple choice question and the second was an open-ended question.

The participants were asked to choose as many as they find fit from the following seven choices: (1) presence of code snippet, (2) presence of url(s) to the materials supporting the opinion, (3) user profile (reputation, activity), (4) number of votes, (5) length of the post, (6) date of the opinion, (7) other. The multiple choices were determined by inspecting the posts of Stack Overflow. Figure 8 presents the results.

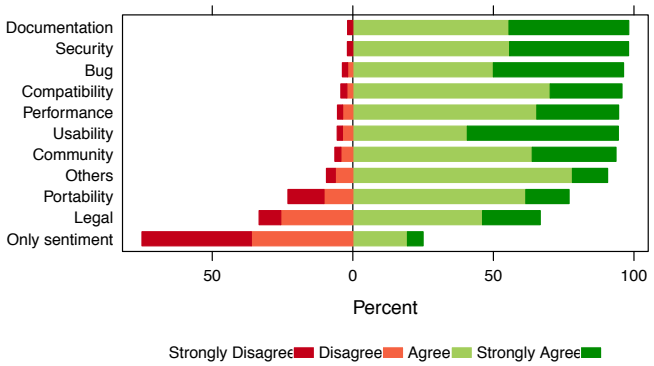

Fig. 7: API aspects to be included in opinion summaries.

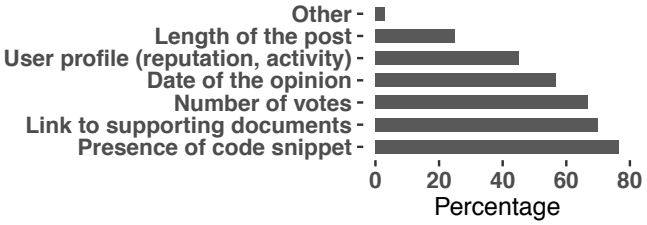

Fig. 8: Factors influencing the quality of API opinions.

Most of the participants (76.7%) consider an opinion, accompanied by a code example, to be of high quality. 70% of the participants find that the presence of links to supporting documents is a good indicator of the quality, while 56.7% believe that the posting date is an important factor. Stack Overflow uses upvotes and downvotes of a post as a gauge of the public's opinion of the content. 66.7% of developers attribute a higher count of votes on the post to an opinion of higher quality. As also found by Vasilescu et al. [29], the reward system in Stack Overflow motivates developers to write useful posts and to engage. The user profile, another Stack Overflow feature, is found by 45% of participants to be useful when evaluating the quality of the expressed opinion. The length of the post is viewed as not contributing to the opinion quality (according to 25% of 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.

The open-ended question asked developers to write about other factors that determine the quality of the provided opinion. Once again, the major category is associated with the documentation quality, since forum posts are considered as informal documentation. We found that developers look for the following attributes when evaluating the opinion quality:

1) **Clarity**: To R19 *"is the post a quick and dirty writing or the person took time in writing the post?"*; while R29 looks for the *"simplicity in the language used"*.
2) **Facts**: Presentation of the information using supporting facts, e.g., *"accurate technical terms"* (R29), using *"screenshots or similar suggestions"* (R25), or *"examples from real world applications"* (R23). To R53 the writing must have *"authoritative tone, not argumentative"*.
3) **Bias**: the opinion must be free from bias, demonstrating *"apparent fairness and weighing pros/cons"* (R40).
4) **Brevity**: *"summary first, in-depth discussion follows (with code examples)"* (R46). R60 elaborates further — *"the post is understandable and well described, it doesn't need to be long, but a quick concise answer that gives the knowledge needed to come up with the answer or gives a detained answer is always beneficial"*.
5) **Context**: *"I'm not sure I'd care for opinions stripped of the context"* (R32).

R57 also evaluates the author of the post — *"...I usually try to get a feel for the experience level of the poster through their writing style and what things they emphasize or ignore"*. Developers also take into account whether the opinion was provided on behalf of an organization. According to R32, *"you can read a lot into tone-of-voice, thought organization, flaming, open-mindedness, employer, even grammar"*.

RQ3: Developers look for the presence of code snippets, links to supporting documentation, upvotes, opinion date and user reputation when evaluating the quality of opinions. Opinions of high-quality are viewed as clear, short and to the point, bias free, and supported by facts.

3.4 Tool Support for Automated API Opinion Analysis (RQ4)

We asked developers two questions about their preferred choice of tools to support automated analysis of API opinions from online developer forums. The first 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 [30]. 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 is the competing APIs tool (6), followed by an opinion miner (1) and an API comparator (4) (Figure 9). The sentiment analyzer (2) is the least desired tool, i.e., developers want tools that offer more context.

Via the open-ended question, we wanted to learn about other tool support, perhaps specific to APIs, that developers desire. During the analysis of an open-ended question, 11 categories emerged. Three of those categories are related to three types of opinion summarization: (1) extractive and adaptive summary, (2) summarize by time/age, (3) categorize by aspects.

Compared to the summarization by aspects where the opinions are grouped by aspects, extractive summary is produced by selecting a subset of existing words and sentences, thereby producing a highlight in a small paragraph. R57 mentions *"I'd like to see an extracted summary of the issues (not so much the opinions) with formalized brief spaces for rebuttals and augmentation ... Basically don't do it all free-form, but don't rely purely on machine-learning either"*. *Summarization by time* is desired,

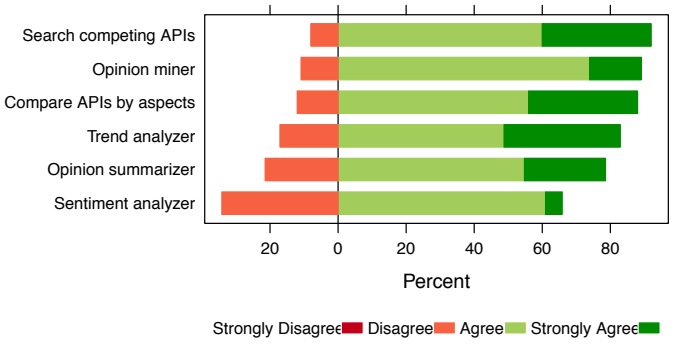

Fig. 9: Tool support for analyzing API opinions.

"undated data is useless in a fast-moving field . . . present by age (last month, last 6 months, . . .)" (R32).

Developers need to see reasoning in the opinions, *"I'd like to see somewhat formalized analyses of the pros and cons – as in, instead of relying only on free form text . . . "* (R57). R23 looks for unusual findings, *"if I frequently find surprising or unusual things, I might discuss those aspects with other users in order to understand if this is a smell or a side effect . . . "*.

Developers want to have traceability between opinion and the corresponding API documents, such as, *"auto link to definitive technical documentation"*, to formal document (R37), to practitioner blogs (R15), and to the client applications where the API is being used. R23 explains how such usage traceability can work, *"I look at the APIs themselves first, then I look at client implementations, etc. I rate these based on the experience and understanding."* R1 wants opinions to be linked to the corresponding API usage history, *"per-project information about when an API has been replaced by if one API got dropped and replaced by another one."*

R39 needs tool support to determine the popularity and maturity of the discussed APIs, which can then be used as a metric to select an API, *"it can be valuable when assessing popularity, as one factor (of many !) when choosing an API"*. Another dimension is *user engagement* in the API discussion. R25 wishes to *"type an API name and see how many questions there are in Stack Overflow divided by the amount of users."*. Grouping of opinions by the user expertise was another theme, *"a programmer with only a few years of experience will have vastly different opinions about an API than someone with a couple decades of experience"* (R13).

RQ4: While developers are interested in a tool support for mining and summarizing opinions about APIs, they also want to link such opinions to other related dimensions, such as, usage, maturity, documentation, and user expertise.

4 Discussion

We discuss the threats to validity (Section 4.1) and the major themes emerged from our study and how those fit into the existing research (Section 4.2). We then describe a prototype tool that we implemented based on themes emerged from the study (Section 4.3).

4.1 Threats and Limitations

The accuracy of the open coding of the survey responses 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 3. While we achieved a high level of agreement, we, nevertheless, share the complete survey responses online: http://bit.ly/2bGPz3H.

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 overrepresented. To mitigate this, we did not place any restriction on the developers' knowledge or experience to be eligible to participate in the survey.

4.2 Implications

Multi-faceted analysis of API opinions. As we observed, the analysis of API-related opinions is often multi-faceted, correlating both technical (e.g., API features) and social aspects (e.g., community engagement). By understanding how opinions guide developers' decisions, we can design tools that can better support their use and development of APIs. Such analysis can add another dimension to the understanding of team productivity through emotion analysis [7], [8].

Augmenting API documentation with opinions. Developers mentioned that opinions about different API elements can help them better understand the benefits and shortcomings of the API. Summaries extracted from Stack Overflow can be effective about the purpose and use of API elements (classes, methods, etc.) [31]. And while the official API documentation still lacks completeness, clarity, and context [27], [32], our results suggest to augment it 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 [33]. Such tools can be integrated within IDEs to assist developers during their APIs-based tasks.

Evaluating benefits of API opinion summaries. While the majority of the developers agreed that opinion summaries can help them with evaluating APIs, three types of summarization were found to be very useful: 1) categorization of opinions by API aspects; 2) opinions summarized as topics; and 3) most important 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. [34] found no clear winner for consumer product summarization, while aspect-based summarization is predominantly present in camera and phone reviews [30], and topic-based summarization has been studied in many domains (e.g., identification of popular topics in bug report detection [35], software traceability [36], etc.).

Measuring opinion quality. Developers expressed their concerns about the trustworthiness of the provided opinions, in particular those that contain strong bias. Since by the definition an "opinion" is a "personal view, belief, judgement, attitude" [37], 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 [30], 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. Such findings can usher a new breed of recommender systems for software engineering [38].

Modelling opinion evolution. Since frameworks and libraries change their APIs [39], opinions about APIs evolve too. Several developers mentioned that they want to be able to track changes in developers' opinions over time. Such needs motivate the design of the model that can represent the opinion evolution, as well as visualization tools that can present such model. An opinion version control system was mentioned as a possible solution. The understanding of the semantic differences between opinions may require linking those to the evolution of APIs and their protocols, possibly presenting an intriguing research direction to understand the barriers developers face while learning to use an API [40].

API expertise recommendation. Our survey participants seek opinions from the experts and consider such opinions as "trusted". The identification of domain experts is an active research area [41], [42], [43]. Research on recommending API experts from the developer forum posts would help developers seeking and evaluating opinions. As we found in the survey, the specific facets (e.g., developer profile and reputation) of the developer forums such as Stack Overflow may be useful.

4.3 Tool Support

Based on the themes and developer needs that emerged from the study, we developed a prototype tool. We are currently working on the final version of the tool and designing experiments to evaluate its performance and usefulness for software developers.

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 current version of the tool mines Stack Overflow forum posts to automatically mine and summarize opinions about API. The tool supports the following features:

- **Opinion mining**: automatically detects positive and negative opinions about the APIs mentioned in the Stack Overflow posts. The tool is modularized, so it can be further extended to mine any posts from any other forums where APIs are discussed.
- **Opinion traceability**: Positive and negative opinions are highlighted in the tool for each API. Each opinion is automatically linked to the forums post where it is found. Developers can hence access forum posts containing an opinion by simply clicking on it.
- **Documentation traceability**: Each API mentioned in the Stack Overflow forum post is automatically highlighted using a custom extension to the Chrome browser. Such highlight is further annotated with a clickable link to the documentation of the API. If a developer while using the tool, visits a Stack Overflow post where an opinion is provided, the extension will be automatically enabled for the post.
- **Opinion ranking**: Each opinion is ranked based on time, i.e., the most recent opinion is presented first. Based on the themes emerged from the study on opinion quality analysis, we are currently investigating on different ranking algorithms to display and filter the most important opinions.
- **Most reviewed APIs**: The landing page of the tool shows the most positively reviewed APIs for a given task. A task is determined as a combination of tags (e.g., 'java+json' corresponds to the json-based tasks using the Java APIs).
- **Opinion summarization**: Based on the summarization needs emerged from the study, we summarized the opinions about each API. The tool currently offers the following four types of summarization for each API:
  1) **Sentiment aggregation**: We detect positive and negative sentiment at the level of each sentence in a forum post for a given API and provide a visual overview of the overall sentiments towards the API.
  2) **Temporal analysis**: We present visual overview of the opinions as provided over time for each API. This overview should help developers to determine how positivity or negativity towards an API have changed over time.
  3) **Topical analysis**: We applied topic modelling algorithm on the positive and negative opinions for each API and grouped the corresponding opinions under each topic.
  4) **Category analysis**: Based on the preferences provided by survey participants on how opinions can be summarized based on categories (e.g., performance, security), we developed a opinion categorization algorithm that automatically categorizes each opinion under a category. The tool shows an overview of the coverage of each category for a given API, as well as allows deep dive into each category by showing which opinions are grouped under each category. We are currently working on a recommendation engine that can offer insights, such as, for a given category (e.g., performance) and for a given task (e.g., json parsing in Java) what would be the most positively or negatively reviewed API?
- **Coherency analysis**: 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.

5 Related Work

Related work can be categorized into four areas. (1) Analysis of developer forums; (2) Sentiment analysis in software engineering; (3) Analysis of APIs in informal documentation; and (4) Summarization of software artifacts.

5.1 Analysis of Developer Forums

Online developer forums have been studied extensively, e.g., to find dominant discussion topics [44], [45], to analyze the quality of posts and their roles in the Q&A process [29], [46], [47], [48], [49], [50], to analyze developer profiles (e.g., personality traits of the most and low reputed users) [51], [52], or to determine the influence of badges in Stack Overflow [53]. Several tools have been developed utilizing the knowledge from the forums, such as autocomment assistance [54], collaborative problem solving [55], [56], and tag prediction [57].

5.2 Sentiment Analysis in Software Engineering

Ortu et al. [7] 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. Mika et al. [8] correlated VAD (Valence, Arousal, Dominance) scores [58] 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. [59] found that security-related discussions in GitHub contained more negative comments. Guzman et al. [60] 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 [10] summarized emotions expressed across collaboration artifacts in a software team (bug reports, etc.) using LDA [61] and sentiment analysis. The team leads found the summaries to be useful, but less informative. Murgia et al. [11] labelled comments from Jira issues using Parrot's framework [62]. They observed only 46.11±5% agreement of the comments where the raters agreed on the same rating, and even lesser agreement when more contexts were added to the comment. Jongeling et al. [63] compared four sentiment tools on comments posted in Jira: SentiStrength [64], Alchemy [65], Stanford NLP [66], and NLTK [67]. While NLTK and SentiStrength showed better accuracy, there were little agreements between the two tools. Novielli et al. [12] analyzed the sentiment scores from the StentiStrength in Stack Overflow posts. They observed that developers express emotions towards the technology, not towards other developers. They found that the tool was unable to detect domain-dependent sentiment words (e.g., bug). These findings indicate that the mere application of an off-the-shelf tool may be insufficient to capture the complex sentiment found in the software artifacts.

5.3 APIs in Informal Documentation

Parnin et al. [68] 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. [50] 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 [69], [70] investigated the relationship between API changes and developer discussions. Our findings contribute to the existing body of knowledge on the topic and suggest future extension of the existing research on API traceability, such as, tracing API code terms in documentation [15], [16], [71], [72], and in emails [17].

5.4 Summarization in Software Engineering

Natural language summaries have been investigated for software documentation [18], [73], [74], [75] and source code [31], [76], [77], [78]. Murphy [79] proposed two techniques to produce structural summary of source code. Storey et al. [80] analyzed the impact of tags and annotations in source code. The summarization of source code element names (types, methods) has been investigated by leveraging both structural and lexical information of the source code [81], by focusing on contents and call graphs of Java classes [77], based on the identifier and variable names in methods [76]. A more recent work proposed a technique to generate automatic documentation via the source code summarization of method context [78]. The notion of context is very important for developers because it helps understand why code elements exist and the rationale behind their usage in the software [32], [82], [83]. The selection and presentation of source code summaries have been explored through developer interviews [84], as well as eye tracking experiment [85]. Our findings confirm that developers also require different types of API opinion summaries.

6 Conclusion

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 a survey with professional developers. 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 found that developers feel frustrated with the amount of available API opinions and 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. We also found that while API opinion summaries can help developers evaluate available opinions, establishing traceability links between opinions and other related documentation can further support their API-related tasks and decisions. The findings and insights gained from this study helped us to build a prototype tool that can help software developers mine and summarize opinions about APIs in a fully automatic way. We are currently refining our tool implementation and designing empirical and user studies with the goal of evaluating performance and usefulness of the tool in practical settings.

Acknowledgements

We thank all developers who took part in our survey for their time, participation, and feedback.

References

[1] github.com, https://github.com/, 2013.
[2] Ohloh.net, https://www.openhub.net/, 2013.
[3] Sourceforge.net, http://sourceforge.net/, 2013.
[4] freecode.com, http://freecode.com/, 2013.
[5] FasterXML, *Jackson*, https://github.com/FasterXML/jackson, 2016.
[6] Google, *Gson*, https://github.com/google/gson, 2016.
[7] 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.
[8] M. Mäntylä, 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] S. Overflow, *Stack Exchange Data Dump*, https://archive.org/details/stackexchange, 2014.

[21] 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.

[22] A. Bacchelli, T. D. Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in *Proc. 34th IEEE/ACM International Conference on Software Engineering*, 2012, pp. 375–385.

[23] 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.

[24] M. Miles and A. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE Publications, 1994.

[25] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[26] D. Freelon, "ReCal2: Reliability for 2 coders," http://dfreelon.org/utils/recalfront/recal2/, 2016.

[27] G. Uddin and M. P. Robillard, "How api documentation fails," *IEEE Softawre*, vol. 32, no. 4, pp. 76–83, 2015.

[28] M. P. Robillard, "What makes APIs hard to learn? Answers from developers," *IEEE Software*, vol. 26, no. 6, pp. 26–34, 2009.

[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] B. Liu, *Sentiment Analysis and Subjectivity*, 2nd ed. Boca Raton, FL: CRC Press, Taylor and Francis Group, 2010.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] Dictionary.com, "Opinion," http://www.dictionary.com/browse/opinion, 2016.

[38] M. P. Robillard, R. J. Walker, and T. Zimmermann, "Recommendation systems for software engineering," *IEEE Software*, vol. 27, no. 4, pp. 80–86, 2010.

[39] D. Dig and R. Johnson, "How do apis evolve? a story of refactoring," *J. Softw. Maint. Evol.*, vol. 18, no. 2, pp. 83–107, mar 2006.

[40] J. Sushine, J. D. Herbsleb, and J. Aldrich, "Searching the state space: a qualitative study of api protocol usability," in *Proceedings 23rd International Conference on Program Comprehension*, 2015, pp. 82–93.

[41] A. Mockus and J. D. Herbsleb, "Expertise browser: A quantitative approach to identifying expertise," in *Proc. 24th International Conference on Software Engineering*, 2002, pp. 503–512.

[42] B. V. Hanrahan, G. Convertino, and L. Nelson, "Modeling problem difficulty and expertise in stackoverflow," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*, ser. CSCW '12, 2012, pp. 91–94.

[43] F. Riahi, Z. Zolaktaf, M. Shafiei, and E. Milios, "Finding expert users in community question answering," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12 Companion, 2012, pp. 791–798.

[44] 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.

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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*, 2013, pp. 460–463.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] Y. Tausczik, A. Kittur, and R. Kraut, "Collaborative problem solving: A study of math overflow," in *In Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2014, pp. 355–367.

[57] C. Stanley and M. D. Byrne, "Predicting tags for stackoverflow posts," in *In Proceedings of the 12th International Conference on Cognitive Modelling*, 2013, pp. 414–419.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] W. G. Parrott, *Emotions in Social Psychology*. Psychology Press, 2001.

[63] R. Jongeling, S. Datta, and A. Serebrenik, "Choosing your weapons: On sentiment analysis tools for software engineering research," in *Proceedings of the 31st International Confernece on Software Maintenance and Evolution*, 2015.

[64] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment in short strength detection informal text," *Journal of the American*

Society for Information Science and Technology, vol. 61, no. 12, pp. 2544–2558, 2010.

[65] IBM, *Alchemy sentiment detection*, http://www.alchemyapi.com/products/alchemylanguage/sentiment-analysis, 2016.

[66] R. Socher, A. Perelygin, J. Wu, C. Manning, A. Ng, and J. Chuang, "Recursive models for semantic compositionality over a sentiment treebank," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013, p. 12.

[67] NLTK, *Sentiment Analysis*, http://www.nltk.org/howto/sentiment.html, 2016.

[68] 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.

[69] 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.

[70] 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)*, 2015, pp. 321–330.

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

[72] N. Bettenburg, S. Thomas, and A. Hassan, "Using fuzzy code search to link code fragments in discussions to source code," in *European Conference on Software Maintenance and Reengineering*, 2012, pp. 319–328.

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

[74] 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, 2006.

[75] 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, 2011.

[76] 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.

[77] 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.

[78] 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.

[79] G. Murphy, *Lightweight Structural Summarization as an Aid to Software Evolution*. University of Washington, 1996.

[80] M.-A. Storey, L.-T. Cheng, I. Bull, and P. Rigby, "Shared waypoints and social tagging to support collaboration in software development," in *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, 2006, pp. 195–198.

[81] 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.

[82] J. Sillito and G. C. M. K. D. Volder, "Asking and answering questions during a program change task," *Journal of IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 434–451, 2008.

[83] S. E. Sim, C. L. A. Clarke, and R. C. Holt, "Archetypal source code searches: a survey of software developers and maintainers," in *International Workshop on Program Comprehension*, Jun 1998, pp. 180–187.

[84] 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.

[85] 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.