Yi-Xuan Tang, Zhi-Lei Ren, He Jiang *et al.* Journal of computer science and technology: Instruction for authors. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 1–25 November 2019. DOI

An Empirical Comparison between API Tutorials and API Crowd Documentation

Yi-Xuan Tang[1], Zhi-Lei Ren[1,*], He Jiang[1,2,3], Xiao-Chen Li[1], and Wei-Qiang Kong[1]

[1] *School of Software, Dalian University of Technology, Dalian 116000, China*

[2] *Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116000, China*

[3] *School of Computer Science & Technology, Beijing Institute of Technology, Beijing 100000, China*

E-mail: tangyixuan@mail.dlut.edu.cn; zren@dlut.edu.cn; jianghe@dlut.edu.cn; li1989@mail.dlut.edu.cn; wqkong@dlut.edu.cn

Received November **, 2019; revised November **, 2019.

Abstract API documentation is critical for developers to learn APIs. However, it is unclear whether different types of API documentation have distinct effects on the API learnability for developers. Thus, we conduct an empirical study to compare the API learnability between official API tutorials and API crowd documentation based on three aspects, i.e., the coverage of APIs, the concerns for APIs, and the presentations. Based on the quantitative analyses, we find that: (i) API crowd documentation can be regarded as a supplement to the official API tutorials in some extent; (ii) the concerns for frequently used APIs between official API tutorials and API crowd documentation show a huge mismatch, which may prevent developers from deeply understanding the usages of APIs through only one type of API documentation; (iii) official API tutorials can help developers seek API information on a long page and API crowd documentation could provide long codes for a particular programming task. These findings may help developers determine which API documentation could be selected for learning APIs and find the useful information they need.

Keywords API documentation, empirical study, quantitative analysis

1 Introduction

Application Programming Interfaces (APIs) provide access to sophisticated functionalities for building and integrating application software. They allow developers to use complex functions without having to modify or understand the underlying implementation details [1]. Thus, software developers tend to reuse APIs in existing frameworks and libraries to facilitate the development process [2, 3]. However, APIs are usually complex and hard to learn, because they may contain hundreds of elements at different levels (packages, types, and methods) and elaborate dependencies [4]. Thus, various types of API documentation provide the behavior of API elements to assist developers in learning APIs, such as API specifications, API tutorials, API crowd documentation, wikis, and blogs [1].

However, it is unclear whether there are any differences on the API learnability between different types of API documentation. API learnability usually refers to how easy an API is to learn [5]. Specifically, we differentiate the official API tutorials and the API crowd documentation as two types of API documentation in

Regular Paper

This work is supported by the National Key Research and Development Program of China under Grants 2018YFB1003900, the National Natural Science Foundation of China under Grants No. 61722202, 61772107 and 61572097, and the Fundamental Research Funds for the Central Universities under Grant No. DUT18JC08.

*Corresponding Author

this paper. Official API tutorials are usually organized by API designers, whereas API crowd documentation is derived from social media, such as Stack Overflow. Although the providers are different for these two types of API documentation, both of them provide explanatory descriptions and code snippets to demonstrate how a certain functionality can be implemented, which is also known as API usage scenarios [6]. Notably, official API tutorials usually contain functional descriptions of API features, which are more authoritative and representative than other API tutorials [2]. Thus, our goal is to examine whether there is a difference and to what extent of the difference on the API learnability between official API tutorials and API crowd documentation.

To meet our objectives, we conduct an empirical study on the Java API documentation and the Android API documentation. As for Java API documentation, we collect 1,062 documents for the Java official tutorial from the Oracle website and 462,154 Java crowd documents based on the ⟨*java*⟩ tag from the Stack Exchange Data Dump. Simultaneously, in order to constitute Android API documentation, we obtain 181 documents for the Android official tutorial from the Android SDK and 393,812 Android crowd documents based on the tag of ⟨*android*⟩. Specifically, we compare these two types of API documentation from three aspects, including the coverage of APIs, the concerns for APIs, and the presentations. The API coverage could determine whether developers can refer to official API tutorials or API crowd documentation to find API information for the specific APIs. The concerns for APIs could reveal whether the official API tutorials can be effectively utilized for learning APIs. The presentations of API documentation could influence the API learnability to some extent. Accordingly, we propose the following research questions, which each question corresponds to one of the aspects:

• **RQ1. What is the coverage of APIs in official API tutorials and API crowd documentation?** We employ a traceability recovering tool to link code-like terms in official API tutorials and API crowd documentation to specific APIs, i.e., Java Standard Edition (SE) APIs version 7* and Android API library version 7.0†. Based on the specific API elements, we can measure the coverage of Java APIs and Android APIs, and determine whether there is a difference on API coverage between official API tutorials and API crowd documentation.

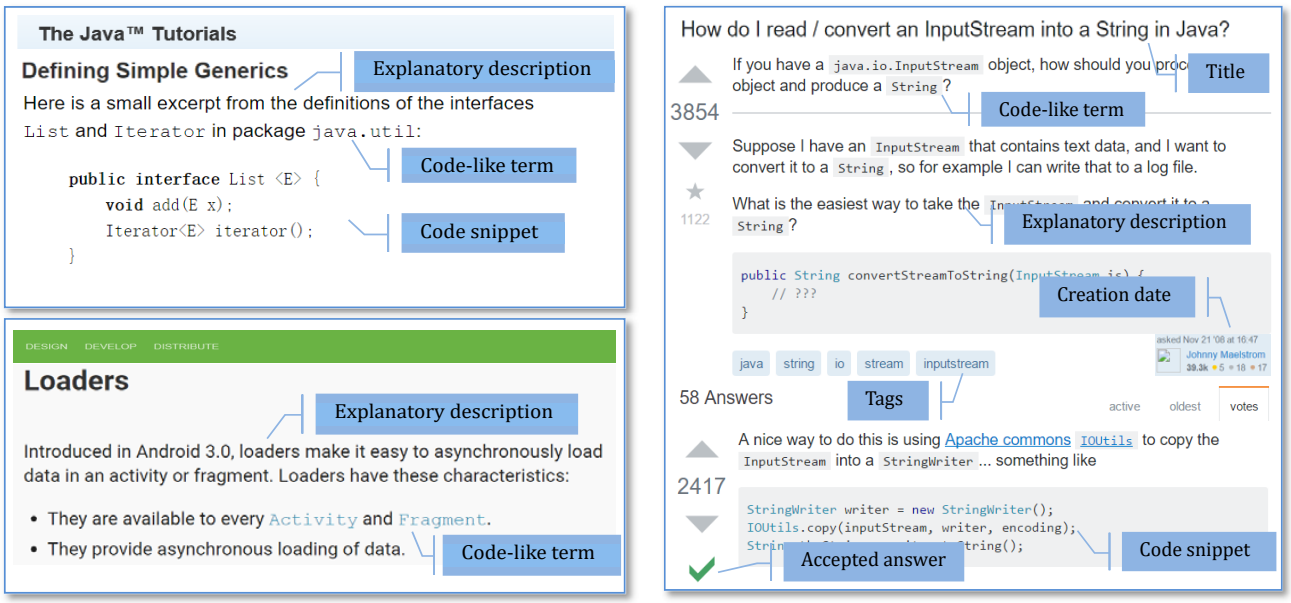
• **RQ2. Are the concerns for APIs consistent in official API tutorials and API crowd documentation?** We treat the ranking orders or the ranking proportions as an indicator of the concerns for APIs in official API tutorials and API crowd documentation. First, we rank the API elements according to their frequencies in the API documentation. Then, we calculate the ranking orders and the ranking proportions for several frequently used APIs to examine whether these APIs are regarded as equally important in these two types of API documentation.

• **RQ3. Are there any differences of the presentation between official API tutorials and API crowd documentation? And to what extent of the differences?** We first conduct a survey to understand what characteristics of API documentation could facilitate API learnability. According to the results of the survey, we then employ the Wilcoxon rank-sum test with effect sizes to differentiate the presentation of official API tutorials and API crowd documentation.

Overall, we discover that the coverage of APIs in API crowd documentation is much higher than that in

*https://docs.oracle.com/javase/7/docs/api/

†https://developer.android.google.cn/reference/packages



(a) (b)

Fig.1. Document examples in official API tutorials and API crowd documentation. (a) Document in the Java tutorial and the Android tutorial. (b) Document in the java crowd documentation.

official API tutorials. Thus, for most APIs, developers can rely on API crowd documentation to learn how to use these APIs. We also find that there are significant inconsistencies on API concerns for the most frequently used APIs. In other words, API designers and developers may focus on different APIs. In addition, official API tutorials contain more explanatory descriptions and hyperlinks regarding an API, which is helpful for developers to learn APIs thoroughly. API crowd documentation is practical for accomplishing programming tasks due to the more quantitative and longer code snippets.

The main contributions of this paper are summarized as follows:

• We provide a systematic and comprehensive empirical comparison between official API tutorials and API crowd documentation. The procedures can also be conducted to compare other types of API documentation.

• We provide evidence for the differences of API learnability between official API tutorials and API crowd documentation from three aspects, namely the different coverage proportions of APIs, the inconsistent concerns for APIs, and the distinct presentations. These differences may help developers select the suitable API documentation to find the desired information they need.

The rest of the paper is organized as follows. In Section 2, we present the background and the related work. In Section 3 and Section 4, we present the case study setup and the results of our case study. We list several lessons learned in Section 5 and discuss the threats to validity our findings in Section 6. Finally, we conclude this paper in Section 7.

2 Background and related work

Work related to our study can be divided into two aspects, namely the work on official API documentation and the work on crowd documentation. Particularly, we refer to three levels of APIs in this paper,

i.e., API Packages, API Types, and API Methods. API Methods can be executed to interact with other objects in the application. An API Type can be a class, an interface, an enumeration, or an annotation, which offers a list of API Methods of communication among various components. API Packages provide a large number of API Types which are grouped into different packages according to functionality, such as "java.lang", "java.util", and "java.io".

2.1 Official API documentation

Official API documentation, such as official API tutorials and API specifications, plays an important role in learning APIs [2]. Official API tutorials are usually organized by a series of programming topics [7, 8]; the API specifications usually explain the functionality, parameters, and possible exceptions for APIs. Since this study focuses on the official API tutorials, we present the examples of official API tutorials in Fig. 1(a). From Fig. 1(a), we can see that the explanatory descriptions and the code snippets are two important components of the official API tutorials. The explanatory descriptions explain the considerations on how to use APIs and the code snippets describe a required sequence of API calls for implementing a certain functionality. Actually, official API documentation is created by a few API designers and consulted by a variety of developers [9]. When facing unfamiliar APIs, developers often refer to API tutorials to understand their correct usages, such as which methods to call and which parameters to use for a method.

In most cases, developers hope to quickly understand how to use an API. However, previous work has shown several challenges on effectively seeking information and using APIs. Robillard et al. [4] observed that insufficient code snippets are an obstacle for developers to learn APIs. He also found five important factors that influence the API learnability, i.e., the documentation of intent, code snippets, cookbooks for mapping usage scenarios to API elements, the penetrability of APIs, and the format and the presentation of the documentation [10]. Scaffidi [11] investigated four challenges in learning and using APIs, i.e., inadequate documentation, insufficient orthogonality, inappropriate abstractions, and incompatible assumptions.

These obstacles and factors motivate researchers to focus on various tasks to facilitate APIs' learning efficiency, such as tutorial API fragments recommendation [7, 12], API documentation enhancement [13], quality measurement [14], and error detection [15]. For examples, tedious API tutorials can be divided into several API tutorial fragments to help developers quickly learn unfamiliar APIs [7, 12]; error documentation can be improved to provide reliable information for developers [15]. Based on the efforts of these researchers, official API documentation can be effectively utilized by developers to some extent.

Similar to the previous work, we also focus on the analysis of factors that may influence the API learnability in the official API documentation. The difference is that we compare how different API documentation (official API tutorials and API crowd documentation) could influence the API learnability. In detail, we conduct a quantitative analysis to provide evidence from three aspects, i.e., the coverage of APIs, the concerns for APIs, and the presentations. These three aspects can also be used to compare other types of API documentation.

2.2 Crowd documentation

Crowd documentation is an online discussed resource, which can be derived by developers' contributions through social media [16], such as Stack Overflow. Stack Overflow, a popular online technical Ques-

tion and Answer (Q&A) forum, allows millions of developers to communicate API-related questions [16, 17]. Thus, abundant API usage information on Stack Overflow forms online crowd documentation.

In Stack Overflow, when a developer meets an API usage question, he/she can check whether there are similar questions that have been solved with the best answers. Once not, a new question can be posted on Stack Overflow. The question is required to be labeled with one to maximum of five tags to categorize it. For example, a Java question with the title of "how to read/convert an InputStream into a String in java" is submitted by a developer in November 2008, as shown in Fig. 1(b). The curating activities (such as voting) and incentives (such as reputation scores) on Stack Overflow can motivate developers to answer this question. In such a way, submitters can get very fast responses in median of 11 minutes [18]. After verifying the responses, the developer can mark one of them as the best answer to the question. The best answer is presented with a green checkmark on the left side of the answer. In order to guarantee the quality of the crowd documentation, we combine the questions and the corresponding best answers into question-answer pairs and select those pairs that contain at least one API element to compose the API crowd documentation.

Previous work on crowd documentation analysis focuses on various aspects, including the coverage of APIs [16], the influence of several code properties on the number of crowd documentation [19], the duplicate questions [20], and the trends and topics in Stack Overflow [21, 22, 23]. Parnin et al. [16] found that the crowds can generate abundant contents with code snippets and achieve the high API coverage of three popular APIs, namely Android APIs, GWT APIs, and Java APIs. For example, the crowd documentation has covered 87% of the Android classes based on the Exchange Data Dump until December 2011. Beyer et al. [19] revealed that the code properties of code metrics RFC (Response For a Class) and WMC (Weighted Method Count) impact the number of crowd documents for Android APIs. Improving these properties can also improve the usability of APIs and reduce the amount of crowd documentation [19]. Zhang et al. [20] proposed an automated approach, namely DupPredictor, to identify duplicate questions on Stack Overflow. The DupPredictor cannot only save resources for developers to answer other questions, but also accelerate the process for solving questions. Yang et al. [21] leveraged a topic model, LDA (Latent Dirichlet Allocation), to cluster different security-related questions on Stack Overflow. They found that the most popular topic was web security. Besides, Rosen et al. [22] and Barua et al. [23] also employed a LDA topic model to summarize the mobile-related questions and discovered main topics in Stack Overflow, respectively.

Different from the previous work, we focus on how API crowd documentation could help developers learn APIs. Besides the API coverage analysis, we also detect which APIs are popular for developers and examine how the presentation of API crowd documents could impact the API learnability.

3 Case Study Setup

In this section, we first construct the dataset of official API tutorials and API crowd documentation. Then, we describe the API traceability tool to identify API elements. Last, we introduce several characteristics to analyze the presentation of these two types of API documentation.

3.1 Data Collection

We collect relevant data from official API tutorials and Stack Overflow for analysis. Specifically, we

analyze the API documentation for Java and Android platforms. We choose Java and Android API documentation for empirical studies, since the Java APIs and Android APIs are popular, diverse in nature, and attract a large scale of users.

We collect the crowd documentation from the Stack Exchange Data Dump published in June 2017[‡]. This dump includes the publicly available information of questions and answers, as well as the user information. We download the posts.xml file in this dump because all the needed information of questions and the corresponding answers are included in this file, such as the tags of questions, the explanatory description, and the code snippets. In order to obtain the posts of Java APIs and Android APIs, we extract the Java-related questions and Android-related questions which are tagged with ⟨*java*⟩ and ⟨*android*⟩, respectively. Furthermore, we exclude the ⟨*android*⟩ tag when collecting Java-related questions. Querying the posts.xml file, we obtain 919,608 Java-related questions with 1,429,555 answers from January 2012 to June 2017. Simultaneously, we obtain 900,789 Android-related questions with 1,184,903 answers during the same period. Nevertheless, the quality of these questions and answers in Stack Overflow varies drastically. For example, some answers do not match the questions and even contain rude words [24]. Thus, we follow the previous work [8] to combine the questions with their best answers to generate a series of question-answer pairs. Only these question-answer pairs are remained as the candidate crowd documents. Then, if the candidate crowd document contains at least one API element, we treat it as an API crowd document. In total, we obtain 462,154 Java crowd documents and 393,812 Android documents from the Stack Overflow.

In addition, we download the Java API tutorial for Java SE APIs version 7 from the Oracle website[§], and the Android API tutorial for Android API library version 7.0 from the Android SDK to construct the official tutorial dataset. The examples of contents for the Java tutorial and the Android tutorial are presented in Fig. 1. We can see that the official API tutorials are mainly composed of explanatory descriptions and code snippets, which can provide specific usage scenarios and examples to explain how to use the APIs. The code-like term in the explanatory descriptions have higher probabilities related to an API element, such as *"runOnUiThread()"*, *"InputStream"*, *"list"*, and *"Iterator"*. In total, we collect 1,062 tutorial documents for Java APIs and 181 tutorial documents for Android APIs for the consequent analysis.

3.2 API traceability

To perform the analysis of API coverage in official API tutorials and API crowd documentation, we first need to link the code-like terms in these two types of API documentation to specific code elements. For example, the java tutorial mentions at the beginning of the *simple generics* section: "$\cdots$ interface *List* and *Iterator* in package *java.util*", as shown in Fig. 1. We can see that there exist two code-like terms "List" and "Iterator", which refer to APIs of "java.util.List" and "java.util.Iterator", respectively. Despite that a code-like term could refer to different hierarchies when ignoring the context of the term, traceability task for APIs should exactly recover a code-like term to its correct API element.

We follow the previous work [25] to consider API elements as public/protected API Packages, API Types, and API Methods. In order to recover traceability links

[‡]https://archive.org/details/stackexchange.

[§]https://www.oracle.com/technetwork/java/javase/downloads/

between an API element and its learning resources (official API tutorials and API crowd documentation), we employ an API traceability tool called RecoDoc [26]. RecoDoc generates a linked model by a pipeline of filtering heuristics and considers external references, parameters, and context hierarchies to avoid ambiguities when recovering traceability links. Thus, RecoDoc can automatically find code-like terms in explanatory descriptions and code snippets, and then link each code-like term to a fine-grained API.

The traceability links can support various release versions and different levels of APIs. In this study, we select Java Standard Edition (SE) APIs version 7 and Android API library version 7.0 to perform the traceability links. As for Java APIs, we extract all levels of APIs from Java SE 7 API Specification, i.e., API Packages, API Types, and API Methods. In total, we collect 209 API Packages, 3,949 API Types, and 30,497 API Methods for Java APIs. As for Android APIs, we follow the approach in the literature [19] to collect Android API Packages which start with "*android*" and the corresponding sub-types and sub-methods from Android specification. Therefore, we obtain 170 API Packages, 660 API Types, and 8,974 API Methods for Android APIs. These selected APIs are used to link the corresponding documents in official API tutorials and API crowd documentation.

3.3 Characteristics extraction

To examine the difference of API learnability based on the presentation between official API tutorials and API crowd documentation, we design and extract 12 characteristics from these two types of API documentation, which can be further divided into three groups, i.e., the explanatory description group, the code snippet group, and the synthesis group. The explanatory description group measures the structural characteristics of sentences. The code snippet group tries to detect properties in code. The synthesis group calculates the intersections between explanatory descriptions and code snippets.

Before extracting these characteristics, we first need to split the documentation into two parts, i.e., the explanatory descriptions and the code snippets. In official API tutorials, code snippets are embedded in the HTML tags ⟨*div class="codeblock"*⟩...⟨*/div*⟩ or ⟨*pre class="codeblock"*⟩...⟨*/pre*⟩. In API crowd documentation, code snippets can be extracted from the HTML tags ⟨*pre*⟩⟨*code*⟩...⟨*/code*⟩⟨*/pre*⟩. The remaining contents are regarded as explanatory descriptions. Then, we can extract characteristics from the explanatory descriptions and the code snippets, respectively. Furthermore, we clarify the description and the rationality for each characteristic in the following parts.

3.3.1 Explanatory description group

We extract six characteristics from the explanatory descriptions of official API tutorials and API crowd documentation. These characteristics are also used to analyze the structures of text information in previous work [27, 28]. We detail these characteristics as follows:

Number of sentences. This characteristic calculates the number of sentences of the explanatory descriptions. Sentences are usually used to provide description knowledge related to an API, such as the usage scenarios or the considerations on how to use the API. We employ an open tool named LingPipe[¶] to identify each sentence in the API documentation. However, there are a large number of tables in official API tutorials, which can seriously decrease the effectiveness of the LingPipe. For example, there are more than four hundred tables in the Java tutorial. Thus, we extract the contents from

[¶]http://alias-i.com/lingpipe

these tables and treat each line of the table as a sentence. Simultaneously, we still leverage the LingPipe tool to detect sentence boundaries if there are symbols of sentences in tables. In such a way, the explanatory descriptions can be divided into a series of sentences.

Length. This characteristic measures the length of the explanatory descriptions. The length of API documentation has distinct effects on API learning. For example, developers prefer longer explanatory descriptions because it is easy to locate the API information on a long page [10]. We leverage the natural language processing steps to get words from the explanatory descriptions, such as tokenization, stemming, and stop word removal [7]. Then, we accumulate the number of words for each explanatory description.

Entropy. This characteristic measures the average information in the explanatory descriptions, which is widely used in the information theory [29]. If the entropy of a document is lower, developers would not get abundant API information, which may have a negative effect on learning APIs. The entropy of the explanatory descriptions can be calculated as follow:

$$Ent(D) = -\sum_{i=0}^{N-1} P(X_i) log_2 P(X_i) \quad (1)$$

where D represents a document, and X_i stands for a unique word in D. $P(X_i)$ is the probability of the distinct word based on the normalized frequency of this word. N indicates the number of distinct words.

Readability. This characteristic estimates the understandability of the explanatory descriptions, which are widely used to analyze the writing style of documents [28]. In this study, we use the Automated Readability Index (ARI) [30] to analyze to what extent of the difficulty for a reader to understand the documents. The value of the *readability* is calculated as follow:

$$ARI(D) = 4.71 \times (\frac{N_{char}}{N_{word}}) + 0.5 \times (\frac{N_{word}}{N_{sen}}) - 21.43 \quad (2)$$

where D stands for a document, N_{char} is the number of characters, N_{word} is the number of words, and N_{sen} is the number of sentences.

Number of hyperlinks. This characteristic calculates the number of hyperlinks in the explanatory descriptions. Hyperlinks could help readers extend their knowledge and understand APIs. In official API tutorials, we observe that there are many navigational hyperlinks within a document. These hyperlinks are usually used to guide readers to find the information they are interested in, rather than augmenting the description of API usage scenarios. Thus, we exclude these navigational hyperlinks by matching the corresponding HTML tags and calculate the number of remaining hyperlinks for each document in official API tutorials. In API crowd documentation, we include all the hyperlinks since these hyperlinks can help explain an API and increase the possibility of addressing the API-related questions.

Number of code-like terms. This characteristic counts the number of code-like terms in the explanatory descriptions. A code-like term is a series of characters that are associated with a specific API element. These code-like terms are usually used to interpret the detailed usability of an API. We extract code-like terms by matching the HTML tags ⟨*code*⟩...⟨*/code*⟩. Additionally, due to the informal nature of Stack Overflow, we also use regular expressions to identify code-like terms, including the functions with parentheses and camel cases. In such a way, we can thoroughly identify code-like terms in official API tutorials and API crowd documentation.

3.3.2 Code snippet group

We extract five characteristics from code snippets of official API tutorials and API crowd documentation. These characteristics are also partially derived and inspired by previous work [19, 31]. We detail these characteristics as follows:

Number of code snippets. This characteristic detects the number of code snippets for API documentation. Code snippets are essential elements of API learning, which can help developers understand the purposes of APIs, API usage protocols, and API usage contexts [32, 33]. We accumulate the number of code snippets by matching HTML tags.

Lines of annotations. This characteristic measures the lines of annotations in code snippets. Annotations are usually used to describe the parameters of functions and interpret the function of the code. They can help readers quickly understand the code snippets [34]. We treat each line in the code snippet as a sentence, and identify each annotation by checking whether a sentence starts with the annotation symbols *"/*"*, *"*"*, or *"*/"* , or contains the symbol *"//"* without a semicolon behind.

Lines of statements. This characteristic measures the size of code snippets, which usually refers to non-commentary lines in code. Longer code snippets tend to provide more interactions among APIs and be more complete for a specific programming task. In order to calculate lines of statements, we first perform the process of annotation identification to filter out annotations in each code snippet, because these annotations cannot be treated as statements. Then, we follow the study by Jiang et al. [12] to split the code snippets into statements. We split the code snippet at the places of semicolons into several code segments and split the code segments into separate statements at the place of the parentheses. Only the statements without pure whitespace and comments are included in the calculation of *lines of statements* for code snippets.

Lines of "new" statements. This characteristic calculates the number of instantiated objects in code snippets, which is usually used to interpret the usage of the constructed function. The more the number of lines of "new" statements is, the more numbers of classes or instances that the code snippet tends to introduce. We accumulate the value of this characteristic if a "new" statement appears in a code snippet.

Cyclomatic complexity. This characteristic measures the furcated syntax structures and conditional loops in code snippets. The higher the value of cyclomatic complexity, the more difficult it is to understand the code. We follow the method in the literature [35] to calculate the cyclomatic complexity of code snippets based on the control flow graph, which is shown as follow:

$$C(D) = \frac{\sum_i^n E_i - N_i + P_i}{n} \tag{3}$$

where D stands for a document and n represents the number of code snippets. N_i is the number of nodes, E_i is the number of edges, and P_i is the number of connected components in the graph.

3.3.3 Synthesis group

Intersection. This characteristic calculates the number of APIs which appear in both explanatory descriptions and code snippets. The intersections between explanatory descriptions and code snippets could impact the understanding of an API [10]. Since code-like terms in explanatory descriptions are linked to specific API elements during the API traceability process, we can determine whether an API appears in both explanatory descriptions and code snippets.

4 Case Study Results

In this section, we answer three research questions proposed in this paper. Section 4.1 analyzes the coverage proportions of APIs in official API tutorials and API crowd documentation (RQ1). Section 4.2 discusses the consistency of the concerns for frequently used APIs (RQ2). Section 4.3 shows the differences and the degree of the differences for the presentation between official API tutorials and API crowd documentation (RQ3).

4.1 RQ1. What is the coverage of APIs in official API tutorials and API crowd documentation?

Motivation. Official API tutorials and API crowd documentation can help developers finding answers for API-related questions [36]. When one API is covered by either official API tutorials or API crowd documentation, developers can refer to the API documentation to learn how to use the API. Once not, developers may spend much time on API documentation but find nothing. To examine whether we can rely on API documentation to learn APIs and which APIs are covered by official API tutorials and API crowd documentation, we set up this question.

Approach. We recover the traceability links of API elements in two learning resources and calculate the coverage proportions of different levels of APIs to determine the API coverage in these two types of API documentation.

First, we leverage the API traceability tool RecDoc to link code-like terms to specific API elements. Specifically, we recover the traceability links to Java SE APIs version 7 and Android API library version 7.0 on different levels, i.e. API Packages, API Types, and API Methods. Since the frequencies of APIs are analyzed in our study, we only identify whether an API appears in the document rather than whether the document is related to an API. Besides, if a fine-grained level of API is found, we also accumulate the coarse-grained APIs it belongs to. For example, if an API Method "toArray()" is discovered in learning resources, we accumulate the frequencies not only of this API Method, but also of the corresponding API Type 'java.util.ArrayList" and the API Package "java.util".

Second, we label an API as "included" if the frequency of this API is non-zero. Otherwise, the API is labeled as "excluded", which represents that we cannot find the information for that API. That is to say, this API is excluded in either official API tutorials or API crowd documentation. Then, the coverage of different levels of APIs can be defined as the proportion of APIs that are labeled as "included" compared with the size of different levels of APIs. Thus, we can calculate and compare the coverage proportions of different levels of APIs in official API tutorials and API crowd documentation.

Table 1. Coverage of Different Levels of Java APIs in Official API Tutorials and API Crowd Documentation

API Levels	# All APIs	# APIs in Official Tutorials (%)	# APIs in Crowd Documents (%)
API Packages	209	113 (54.07%)	192 (91.87%)
API Types	3,949	1,261 (31.93%)	2,410 (61.03%)
API Methods	30,497	3,789 (12.42%)	8,266 (27.10%)

Table 2. Coverage of Different Levels of Android APIs in Official API Tutorials and API Crowd Documentation

API Levels	# All APIs	# APIs in Official Tutorials (%)	# APIs in Crowd Documents (%)
API Packages	170	68 (40.00%)	163 (95.88%)
API Types	3,456	660 (19.10%)	2,629 (76.07%)
API Methods	29,017	927 (3.19%)	8,974 (30.92%)

Results. Table 1 and Table 2 show the API coverage proportions of Java APIs and Android APIs at different levels in official API tutorials and API crowd documentation, respectively. We can observe that the API coverage proportions in the API crowd documentation is much higher than that in the official API tu-

Table 3. Proportions of APIs covered by either Official API Tutorials or API Crowd Documentation

Language	API Levels	# APIs in both Two API Documentation(%)	# APIs only in API Crowd Documentation(%)	# APIs only in Official API Tutorials(%)	# APIs in None (%)
Java	API Packages	112 (53.59%)	80 (38.28%)	1 (0.48%)	16 (7.65%)
	API Types	1,114 (28.21%)	1,296 (32.82%)	147 (3.72%)	1,392 (35.25%)
	API Methods	2,582 (8.47%)	5,684 (18.64%)	1,207 (3.96%)	21,024 (68.93%)
Android	API Packages	68 (40.00%)	95 (55.88%)	0 (0.00%)	7 (4.12%)
	API Types	642 (18.58%)	1,987 (57.49%)	18 (0.52%)	809 (23.41%)
	API Methods	797 (2.75%)	8,177 (28.18%)	130 (0.45%)	19,913 (68.62%)

torials. For example, the coverage proportion of Java API Types is 61.03% in the Java API crowd documentation, which is nearly twice as large as that in the Java official tutorial. Similarly, in the Android API crowd documentation, the coverage proportion of API Types reaches to 76.07%, while the proportion is only 19.10% in the Android official tutorial. The reason could be that developers can find more intricacies of APIs when calling the APIs in practice and know which APIs have crashed, rather than API designers. Therefore, developers may post more real API usage examples than API designers.

As for different levels of APIs, we can observe that the coarser the level of APIs, the higher the API coverage. For example, the coverage of Java API Packages is 54.07% in the Java official tutorial, whereas the coverage of Java API Methods is only 12.42%. However, developers prefer API documentation that does not focus on the complete information of API Methods, because inappropriate method-level documentation can decrease developer productivity to some extent [10].

Table 3 shows the detailed API coverage proportions in either official API tutorials or API crowd documentation. We can see that the proportion of APIs which are covered by both official API tutorials and API crowd documentation decreases as the level of APIs is refined. However, the proportion of APIs that are not covered by any of the API documentation increases, especially for the Android APIs. We can also find that the proportions of APIs that are covered by only API crowd documentation are much higher than APIs that are covered by only official API tutorials. For example, the proportion of API Packages that are covered by only Java API crowd documentation is 38.28%, while the coverage proportion is 0.48% for the Java official API tutorial. Notably, only one Java API Package "javax.sound.midi.spi" is appeared in the Java tutorial but not discussed in the Java API crowd documentation in our dataset. Besides, no Android API Packages that are covered by only official API tutorials.

Summary. Developers can refer to API crowd documentation to learn how to use APIs due to the high API coverage. Since most APIs that are covered by official API tutorials are also covered by API crowd documentation, API crowd documentation has turned into a worthy extension to help developers learn APIs.

4.2 RQ2. Are the concerns for APIs consistent in official API tutorials and API crowd documentation?

Motivation. Official API tutorials are usually organized by a series of programming topics [7]. When an API is critical in most programming tasks or the usage scenarios of the API need more explanations, API designers would introduce the API in detail. In most cases, the frequency of this API would be higher than other APIs in official API tutorials. However, APIs discussed in API crowd documentation are the real concerns for developers on accomplishing development tasks [16]. If

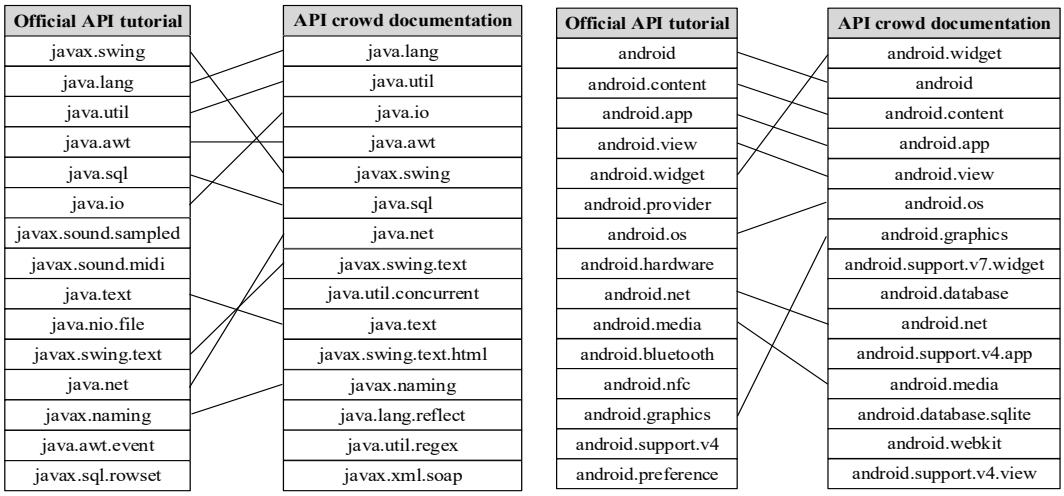


Fig.2. Top 15 API Packages in official API tutorials and API crowd documentation. (a) Java API Packages that are frequently used by API designers vs. developers. (b) Android API Packages that are frequently used by API designers vs. developers.

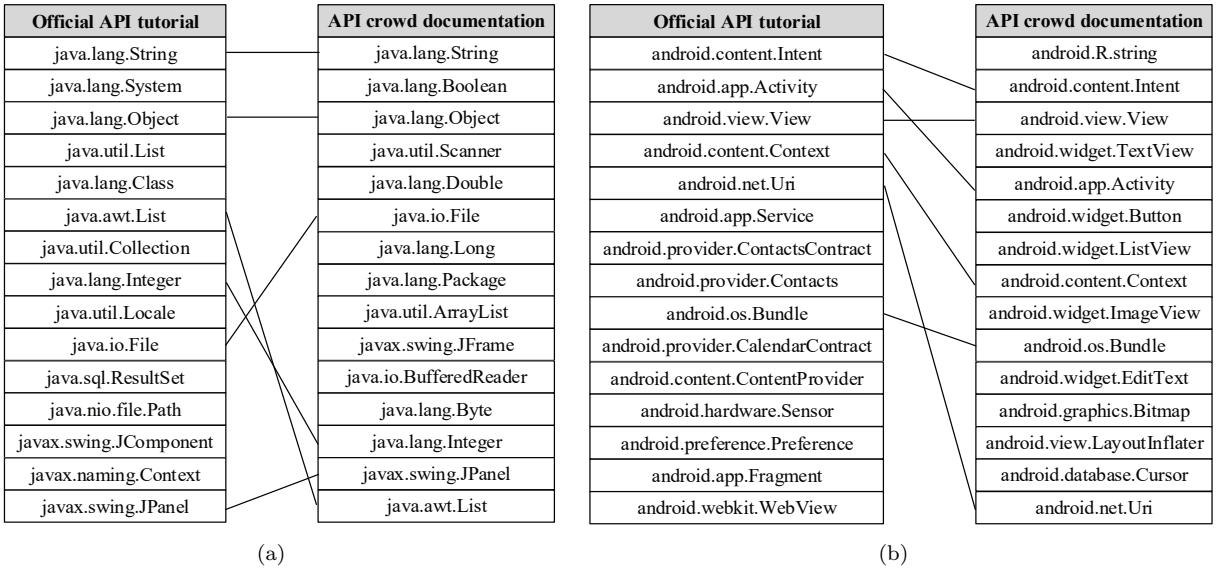


Fig.3. Top 15 API Types in official API tutorials and API crowd documentation. (a) Java API Types that are frequently used by API designers vs. developers. (b) Android API Types that are frequently used by API designers vs. developers.

the concerns of an API show discrepancies between official API tutorials and API crowd documentation, developers may not deeply understand how to use the API though only one API documentation, which may seriously decrease the API learning efficiency. To examine what APIs are critical to developers and what APIs are frequently discussed by API designers, we set up this question.

Approach. We select the top 15 frequently used APIs and the top 10 APIs in large sizes to examine whether there is a mismatch on the concerns for APIs between API designers and developers. Then, we employ the Spearman's rank correlation [37] to evaluate the extent of the mismatch.

First, we rank the frequencies of APIs that appeared in official API tutorials and API crowd documentation,

Table 4. Ranking Proportion for API Packages

Language	API Packages	Official API tutorials (%)	API crowd documentation (%)	Absolute difference (%)
java	**org.omg.CORBA**	**65.38%**	**12.30%**	**53.08%**
	javax.swing	0.96%	4.28%	3.32%
	java.awt	3.86%	9.09%	5.23%
	javax.swing.text	10.58%	25.67%	15.09%
	java.lang	1.92%	0.53%	1.39%
	java.util	2.88%	1.07%	1.81%
	java.security	19.23%	41.18%	21.95%
	java.io	5.77%	1.60%	4.17%
	javax.print.attribute.standard	**100.00%**	**39.57%**	**60.43%**
	javax.management	35.58%	42.25%	6.67%
android	android.provider	9.68%	15.95%	6.27%
	android.widget	8.06%	0.61%	7.45%
	android.media	16.13%	7.36%	8.77%
	android.view	6.45%	3.07%	3.38%
	android.support.v17.leanback.widget	**NA**[a]	**23.93%**	**-**
	android.app	4.84%	2.45%	2.39%
	android.icu.text	**91.94%**	**14.11%**	**77.83%**
	android.graphics	20.97%	4.29%	16.68%
	android.os	11.29%	3.68%	7.61%
	android.support.v7.widget	**NA**[a]	**4.91%**	**-**

Note: NA[a] represents no information for the APIs.

and select the top 15 APIs by descending order to check the consistency of APIs. Second, we calculate the ranking proportion for each API. Notably, APIs are ranked in the same order if they have the same frequencies. As for the APIs that are not used in either official API tutorials or API crowd documentation, we do not calculate the ranking proportions of them. Third, we calculate the absolute difference of ranking proportions for the API. For example, if the ranking proportion of an API "org.omg.CORBA" is 65.38% in the Java tutorial and is 16.04% in the Java crowd documentation, then the absolute difference of ranking proportions of this API is 49.34% ($|65.38\% - 16.04\%|$). In such a way, we can analyze whether there is a mismatch on the concerns for a specific API. Finally, we calculate the Spearman's rank correlation to show the extent of the mismatch.

Results. The APIs that are frequently used by API designers and developers are shown in Fig. 2 and Fig. 3. We only list the top 15 API Packages and API Types due to space restrictions. We believe that these top APIs can provide evidence for the mismatch on the API concern for API designers and developers. In Fig. 2(a), APIs in the left column are ranked decreasingly based on the frequencies in official API tutorials, while APIs in the right column are ranked decreasingly based on the frequencies in API crowd documentation. Lines across columns connect the same APIs that are frequently used by both API designers and developers. We can observe that only 67% Java API Packages and 60% Android API Packages in the top 15 perform relatively consistent between official API tutorials and API crowd documentation in Fig. 2. As for those mismatch APIs, the most notable ones are "javax.sound.midi", "javax.sql.rowset", and "android.nfc", which are frequently appeared in official API tutorials but rarely appeared in API crowd documentation. However, APIs "javax.swing.text.html", "javax.xml.soap", "an-

Table 5. Ranking Proportion for APIs Types

Language	API Packages	Official API tutorials (%)	API crowd documentation (%)	Absolute difference (%)
java	java.awt.Component	0.25%	2.16%	1.91%
	java.sql.ResultSet	1.08%	1.36%	0.28%
	java.sql.DatabaseMetaData	46.76%	21.78%	24.98%
	javax.swing.JTable	3.14%	2.54%	0.60%
	javax.swing.JComponent	1.27%	6.11%	4.84%
	javax.swing.plaf.synth.SynthPainter	**56.18%**	**NA**[a]	**-**
	javax.swing.JTree	11.18%	13.55%	2.37%
	javax.sql.rowset.BaseRowSet	**76.08%**	**NA**[a]	**-**
	javax.swing.plaf.basic.BasicTreeUI	**NA**[a]	**66.98%**	**-**
	javax.sql.RowSet	2.35%	45.91%	43.56%
android	android.view.View	0.57%	0.13%	0.44%
	android.widget.TextView	8.52%	0.17%	8.35%
	android.app.Activity	0.38%	0.21%	0.17%
	android.opengl.GLES20	**100%**	**38.61%**	**61.39%**
	android.opengl.GLES30	100%	55.63%	44.37%
	android.app.admin.DevicePolicyManager	11.55%	16.86%	5.31%
	android.support.v7.widget.RecyclerView.LayoutManager	**NA**[a]	**15.61%**	**-**
	android.view.ViewGroup	5.68%	1.79%	3.89%
	android.content.Intent	0.19%	0.08%	0.11%
	android.opengl.GLES10	83.71%	84.22%	0.51%

Note: NA[a] represents no information for the APIs.

droid.database", and "android.support.v7.widget" are frequently used by developers rather than API designers. In Fig. 3, we can observe that only 40% Java API Types and 40% Android API Types in the top 15 show relatively consistent for developers and API designers.

We present the ranking proportions for 10 API Packages and 10 API Types in Table 4 and Table 5, respectively. These APIs are ranked in descending order according to their sizes. Since the ranking proportions are calculated based on the ranked frequencies of APIs, the smaller ranking proportions indicate that these APIs are more frequently used by API designers or developers. Also, the "NA" represents no information related to an API in our dataset. Thus, we cannot calculate the absolute difference for that API between official API tutorials and API crowd documentation. If the absolute difference exceeds 50%, we think there is a significant difference on the usage of that API between official API tutorials and API crowd documentation. We show them in the bold fonts in Table 4 and Table 5. From the tables, we can observe that most APIs have small ranking proportions in both official API tutorials and API crowd documentation, whereas several APIs show huge absolute differences between API designers and developers, such as "javax.print.attribute.standard", "android.icu.text", "android.support.v7.widget", and "android.opengl.GLES20". We find that these APIs are rarely explained or used by API designers but frequently discussed by developers. In contrast, several APIs which are frequently used by API designers are rarely used by developers, such as "javax.swing.plaf.synth.SynthPainter" and "javax.sql.rowset.BaseRowSet". Although we only list 10 API Packages and 10 API Types, we can still observe the mismatch on the concern for the APIs between API designers and developers.

Besides, the Spearman correlations of API Packages between official API tutorials and API crowd documentation are 0.631 and 0.663, which indicates a strong correlation between the coarse-grained APIs. However, it is far from being ideal for Java API Types and Android API Types since the Spearman correlations are 0.425 and 0.109, respectively. In the Method level, the Spearman correlations are 0.016 and 0.070, which show a huge mismatch. The observation indicates that the APIs provided in official API tutorials might not focus on the APIs that developers need. Therefore, developers may not deeply understand the usages of APIs when merely relying on official API tutorials for learning APIs, which may also decrease the API learnability.

Summary. APIs frequently used by API designers and developers are relatively inconsistent. API designers and developers show different concerns for most APIs, especially for the fine-grained APIs. Hence, it would hinder developers from learning APIs merely though one type of API documentation if there is not desirable API information as developers' need.

4.3 RQ3. Are there any differences of the presentation between official API tutorials and API crowd documentation? And to what extent of the differences?

Motivation. Official API tutorials are usually created by API designers which are organized by a series of programming topics, whereas API crowd documentation is derived by crowd contributions through social media [7, 8]. Different providers and design goals may make the presentations of these two types of API documentation behave differently. Recently, a combination of surveys and in-person interviews shows that the presentation of API documentation may indeed impact the API learnability for developers [10], such as the *number of code snippets*, the *length*, and the *number of hyperlinks*. For example, hyperlinks can benefit experienced developers, because they would augment the API information to help developers understand better [10]. In addition, longer pages can help developers locate the useful API information better than a large number of shorter related pages [10]. Simultaneously, code snippets can provide abundant usage information for APIs, which can facilitate developer productivity [38]. Thus, developers can quickly use these APIs to accomplish development tasks. To investigate whether there are significant differences on the API learnability between official API tutorials and API crowd documentation, we set up this question.

Approach. We conduct a survey to better understand what characteristics the developers prefer in API documentation. Then we analyze the differences and to what extent of the differences of these characteristics by the Wilcoxon rank-sum test [39] and the Spearman's rank correlation.

First, in order to understand what can help developers learn APIs in API documentation, we design both close-ended and open-ended survey questions. Based on the guidelines in the literature [40], we limit answer types to numeric and long free-form text. In particular, we first inquire about the programming experiences of developers, and then design five close-ended questions and one open-ended question in Fig. 4. These questions gather answers for characteristics of the presentation that we defined in Section 3.3. For the close-ended questions, we provide several options (i.e., more and fewer), plus an additional option (i.e., I prefer not to answer). The additional option is provided in survey respondents that have no idea on the question. Following the study by Zou et al. [41], we send survey invitations to both industrial professionals and OSS active developers to get enough respondents. On the one hand, we contact industrial professionals working in different companies, including Microsoft, Huawei, Alibaba, and

A survey on the presentation of the API documentation

Q1. How many years have you been programming?

Answer to Q1: _____

Q2*. How many textual details do you prefer in API documentation to explain APIs?

A. Fewer B. Moderate C. More

Q3*. How many hyperlinks do you prefer in API documentation?

A. Fewer B. Moderate C. More

Q4*. How many code snippets and how long of them do you prefer in API documentation?

A. I prefer longer code and more code snippets.
B. I prefer longer code but fewer code snippets.
C. I prefer shorter code but more code snippets.
D. I prefer shorter code and fewer code snippets.
E. Sorry, I prefer not to answer.

Q5*. How many annotations do you prefer in code snippets?

A. Fewer B. Moderate C. More

Q6*. How important of the correlation do you think between textual details and code snippets is?

A. Very unimportant B. Unimportant C. Neutral D. Important E. Very Important

Q7. According to your experience, what other characteristics do you prefer in API documentation to help understand an API?

Answer to Q7: _____

*: exclusive choice

Fig.4. The questionnaire presented to developers.

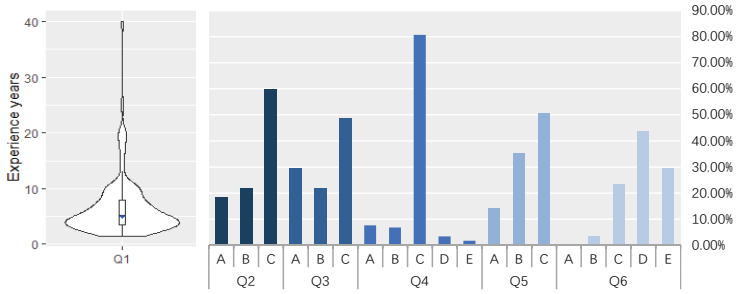


Fig.5. Answers for the questionnaire.

Baidu. They are invited to participate in our survey and help us distribute the survey to their colleagues if possible. On the other hand, we intensively mine 2,000 developers who have submitted commits to the popular Java and Android projects with more than 100 stars on GitHub and send survey invitations to these developers by email. After collecting all the survey responses, we calculate the proportions of options for each question and analyze whether different selected options are influenced by the experience of developers.

Second, we observe the distributions of 12 characteristics which are defined in Section 3.3. The boxplot graphics are employed to present the maximum value, the median value, and the minimum value for each characteristic. Furthermore, we conduct the two-sided Wilcoxon rank-sum test to determine whether it is significantly different for each pair of characteristics. The two-sided Wilcoxon rank-sum test outputs a p-value as the evaluation metric. A p-value below 0.05 indicates statistical significance. Otherwise, we reject the null hypothesis (i.e., the pair of characteristics are from the same distributions) and accept the alternative hypothesis (i.e., the pair of characteristics do not conform to the same distributions). We select the Wilcoxon rank-

sum test since it does not have any assumption on the distribution of the datasets. In addition, we also calculate the rank-biserial correlation to measure the effect size of the Wilcoxon rank-sum test.

Finally, we analyze the correlations of these characteristics within groups and across groups by the Spearman's rank correlation. If the two characteristics have a higher value of correlations, they would present positive correlations and follow a similar trend on the distributions. We choose Spearman's rank correlation since it does not have any assumption on the distributions of the datasets.

Results. In total, we receive 119 responses. The results are summarized in Fig. 5. The left boxplot represents the experience of respondents and the right bar graphic presents the results to other questions. The horizontal axis shows different options of the question regarding the characteristics of the presentation, and the vertical axis shows the proportions of different options received from developers.

The experiences of our respondents vary from 1.5 years to 40 years, with average experience of 6.5 years. Similar to the previous study by David et al. [42], we consider the developers with three levels, including low experience (ExpLow), medium experience (ExpMed), and high experience (ExpHigh). We define the developers with ExpLow as 25% with the least experience in years ($\leq$ 3.5 years in this survey), the developers with ExpHigh as 25% with the most experience in years ($\geq$ 8.0 years in this survey), and the remaining developers with ExpMed that have 3.5 to 8 years of experience.

From Fig. 5, we can observe that almost 50% developers pick "More" options in Q2, Q3, and Q5, while only about 20% developers rate "Fewer" options. In order to gain a better insight into whether one level of developers tend to select "Fewer" or "More" option than other levels of developers, following the literature [41], we conduct Fisher's exact test [43] with Bonferroni correction [44] on these questions from different levels of developers. For the null hypothesis, we assume that different levels of developers tend to select distinct options. After three pairwise comparisons (ExpHigh vs. ExpMed, ExpHigh vs. ExpLow, and ExpMed vs. ExpLow), we find that the p-values are among 0.302 to 1.0 for these questions, indicating that there is not a significant difference (p-value $>$ 0.05) on the selection of opposite options for different levels of developers. Besides, we can observe that more than 60% developers prefer longer documents in Fig. 5. Nearly 50% developers think more hyperlinks and more annotations in code snippets could help understand APIs.

For Q4, more than 80% developers prefer shorter code but more code snippets in the API documentation, especially the developers with ExpMed, which exceed a half of the number. In addition, the proportions of the "important" option and the "very important" option for Q6 are 43.70% and 29.41%, respectively. This means that more than 70% developers think the intersection between explanatory descriptions and code snippets is important for learning APIs.

We receive 59 developer comments for the open-ended questions in the survey responses. Most comments stress the importance of in-depth descriptions, short code snippets, and clear usage caveats. Several developers point out other suggestions on improving API learnability:

Details on how APIs work. *"Simplicity, detail and explanations on what the API returns, and what type of information is returned including the format should be included."*. Another one: *"Test APIs and detailed overview of parameters and returned values can help more."*

Possible errors and usage caveat information. *"It should show the possible flaws, like how it processes*

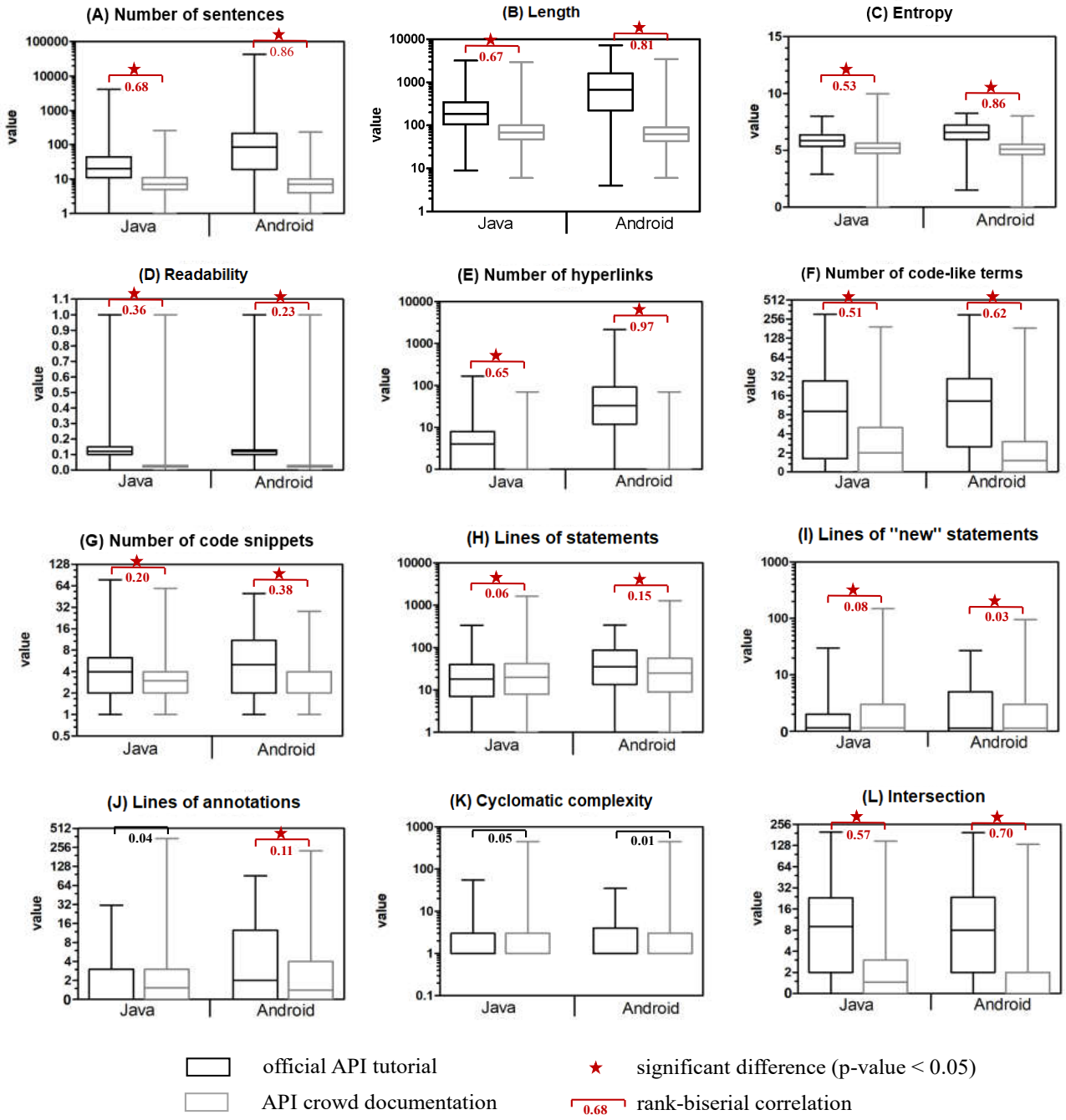


Fig.6. Distribution of different characteristics for presentation in official API tutorials and API crowd documentation.

errors, etc." And "the documentation should allow to copy/paste code that people can try themselves, caveats should be listed."

Code design decisions. *"Something I like to see (and what I try to do as well), is that the developer explains why he/she made certain code design decisions (i.e. what a complex function does backend). Sometimes an API is used by inexperienced developers. Explaining why certain decisions have been made while designing the API can help developers gain more experience."*

Hyperlinks to API's source code or project

using it. *"Sometimes a link to the API's source code if possible."* And *"I expect additional information like 'since', hyperlink to sources, etc."*

We present the distribution for each characteristic as well as the result of Wilcoxon rank-sum tests in Fig. 6. From Fig. 6(A) to Fig. 6(F), we can observe that more than 75% official API tutorials have more sentences, longer documents, and higher entropy values than 75% API crowd documents. With low p-values and high effect sizes, we can find that there are significant differences on the distributions of these characteristics. Since our survey shows that more than 60% developers prefer longer documents with textual details to learn APIs, we can conclude that a long official tutorial page would be more convenient for developers to learn APIs than a dozen short API crowd documents. For the *readability*, we can find that most values concentrate on the lower side of the boxplot graph for both two types of API documentation. This means that it is not difficult for developers to understand the explanatory descriptions. As for the *number of hyperlinks*, 75% API crowd documents do not contain hyperlinks, whereas 75% Android official tutorials have more than 10 hyperlinks in Fig. 6(E). In addition, 50% official API tutorials have more numbers of code-like terms than 75% API crowd documents.

In the code snippet group (Fig. 6(G) - Fig. 6(K)), although the *number of code snippets* and *lines of statements* show a slight difference in these two types of API documentation due to the low p-values, we can also observe that more than 50% official API tutorials have more number of code snippets than 75% API crowd documents. That is to say, official API tutorials have shorter code but more code snippets, whereas most API crowd documents have longer code but fewer code snippets. In addition, the *lines of "new" statements* and the *lines of annotations* also have low p-values with low effect sizes. The values of *cyclomatic complexity* are below 10 for more than 75% code snippets, which indicates that it is not difficult for developers to understand the code snippets in both official API tutorials and API crowd documentation.

In the synthesis group (Fig. 6(L)), we can see that 50% official API tutorials have more intersections than 75% API crowd documents. Simultaneously, the p-values below 0.05 with the effect sizes of 0.57 and 0.70 also present a significant difference. This means that official API tutorials with more intersections between explanatory descriptions and code snippets would be more friendly for developers to learn APIs.

We present the Spearman's rank correlation of these characteristics within groups and across groups in Table 6, 7, and 8. The values more than 0.500 indicate moderate correlations or higher between a pair of characteristics. From Table 6, we can find that the *number of sentencs*, the *length*, and the *entropy* have extremely strong positive correlations due to the high value from 0.669 to 0.983 for both Java and Android API documentation. In the code snippet group, the *lines of statements* present strong correlations with other characteristics as shown in Table 7. The reason may be that longer code snippets tend to contain more instantiate object codes, annotation codes, and furcated syntax structures than shorter code snippets. In Table 8, we can see that the *number of code-like terms* and the *number of code snippets* have strong positive correlations with the *intersection*. Since the intersection is one of the most important characteristics in the API documentation, we can appropriately increase several code snippets for API elements and interpret the usages of these API elements in explanatory descriptions to improve API learnability.

Summary. Official API tutorials have longer documents and more intersections, which are friendly for

Table 6. Spearman's Rank Correlation Within Explanatory Description Group

Spearman's rank correlation	number of sentences	length	entropy	readability	number of hyperlinks	number of code-like terms
number of sentences	-	**0.900/0.746**	**0.669/0.695**	-0.515/-0.327	0.826/0.212	0.457/0.195
length	**0.941/0.781**	-	**0.807/0.903**	-0.262/0.071	0.863/0.285	0.610/0.225
entropy	**0.848/0.723**	**0.906/0.901**	-	-0.106/0.060	0.648/0.290	0.560/0.202
readability	-0.174/-0.208	-0.02/0.159	0.000/0.144	-	-0.214/0.067	-0.010/0.156
number of hyperlinks	0.476/0.213	0.474/0.275	0.471/0.284	0.004/0.114	-	0.404/0.035
number of code-like terms	0.592/0.244	0.604/0.273	0.568/0.226	0.076/0.111	0.361/0.08	-

Note: Left values of "/" represent the correlations of a pair of characteristics in official API tutorials and the right are correlations for API crowd documentation. "-" means no correlations between two characteristics. The values above "-" are measured based on Android API documentation and the below are for Java API documentation.

Table 7. Spearman's Rank Correlation Within Code Snippet Group

Spearman's rank correlation	number of code snippets	lines of code (LOC)	lines of "new" statement	lines of annotation	cyclomatic complexity
number of code snippets	-	**0.897/0.747**	0.650/0.502	0.693/0.540	0.637/0.439
lines of statements	**0.884/0.635**	-	**0.771/0.652**	**0.793/0.733**	**0.720/0.638**
lines of "new" statements	0.575/0.386	**0.664/0.636**	-	0.790/0.534	0.768/0.574
lines of annotations	0.530/0.386	**0.624/0.609**	0.454/0.430	-	0.749/0.517
cyclomatic complexity	0.529/0.315	**0.664/0.589**	0.443/0.469	0.442/0.381	-

Note: Left values of "/" represent the correlations of a pair of characteristics in official API tutorials and the right are correlations for API crowd documentation. "-" means no correlations between two characteristics. The values above "-" are measured based on Android API documentation and the below are for Java API documentation.

Table 8. Spearman's Rank Correlation Across Groups

		Synthesis group	
Groups	Characteristics	Java	Android
Explanatory description group	number of sentences	0.513/0.158	0.318/0.102
	length	0.529/0.170	0.481/0.122
	entropy	0.473/0.147	0.455/0.123
	readability	0.048/0.079	0.153/0.142
	number of hyperlinks	0.239/0.005	0.316/-0.034
	number of code-like terms	**0.821/0.800**	**0.854/0.753**
Code snippet group	number of code snippets	**0.754/0.346**	**0.663/0.361**
	lines of statements	0.772/0.263	0.590/0.316
	lines of "new" statements	0.541/0.189	0.279/0.204
	lines of annotations	0.377/0.163	0.322/0.220
	cyclomatic complexity	0.516/0.156	0.300/0.187

Note: Left values of "/" represents the correlations of a pair of characteristics in official API tutorials and the right are correlation values for API crowd documentation.

developers to seek and learn APIs on a page. API crowd documentation provides longer code snippets for a particular programming task. These benefits can help developers quickly learn APIs and can also be merged into newly API documentation to improve the API learnability to some extent.

5 Lessons learned

Developers might not be able to find sufficient information about uncovered APIs in API documentation. Previous studies have pointed out that insufficient learning resources have been considered as one of the most severe API learning obstacles [4, 10, 11]. Although the official API tutorials and

API crowd documentation achieve the high coverage of APIs as shown in RQ1, there still exist several uncovered APIs without sufficient information, especially for the fine-grained APIs. For example, when a developer wants to seek information about how to use a Java API Type "java.sql.SQLInput", there is no related information about this API in official API tutorials and no best answers to the questions related to this API in API crowd documentation. Unfortunately, if the developer resorts to other types of API documentation but finds nothing, this will seriously delay the development efficiency. Thus, our results are a warning or an incentive to API designers: sufficient API learning resources should be prepared for developers learning APIs.

The mismatch between API documentation would increase the browsing time for finding useful API information. As results shown in RQ2, the frequently used APIs between official API tutorials and API crowd documentation show significant differences. Developers might not get the desirable API information when merely referring to one type of API documentation. For example, when a developer faces an Applet problem, such as "my applet does not display", there is little information guiding how to solve this problem in official API tutorials. However, there are nearly one hundred relevant API crowd documents in our dataset and the number of the crowd documents would increase if we do not limit the searching on Stack Overflow. As a result, developers may not understand how to use the related APIs from official API tutorials but have to go through multiple pages before finding useful API information on Stack Overflow. This phenomenon seriously delays the schedule of developers and decreases the efficiency of API learnability. Thus, our analysis is an incentive to API researchers: practical API information mining tools should be prepared for developers to quickly learn how to use an API or solve the API-related questions. For example, previous work has created a question answering bot for API documentation. It can provide suggestions for developers based on questions that are asked before [36]. Therefore, developers could save much time on browsing API crowd documentation and improve the development efficiency.

The benefits of presentations in distinct API documentation could be merged into newly API documentation to improve the efficiency of API learnability for developers. There are significant differences for most characteristics of the presentation between official API tutorials and API crowd documentation as shown in RQ3. Although the different presentation is due to the distinct design goals of official API tutorials and API crowd documentation, the characteristics of these two types of API documentation that have positive effects on the API learnability also could be integrated into newly created API documentation. For example, API tutorials can be automatically generated from API crowd documentation according to the different complexity of understanding, which can be adaptable to different levels of developer experience [45]. As for the previous API documentation, some tasks aim to optimize the different characteristics of the API documentation. For examples, tedious API documents can be split into relevant tutorial fragments for APIs to accelerate the learning of unfamiliar APIs [7, 12]; code snippets and explanatory descriptions can be extracted from API crowd documentation to enhance the official API documentation [46, 47]. Such work can also improve the quality of the API documentation and optimize the developers' experiences when browsing the API documentation.

6 Threats to validity

6.1 External validity

Threats to external validity are the API bias and dataset limitation.

API bias. We choose two types of APIs, namely the Java SE APIs version 7 and the Android API library version 7.0, to perform our analysis. Other APIs in different programming languages are not analyzed due to space restrictions. However, we believe that other API documentation also exists similar issues for learning APIs. For example, a Python developer sends a mail to us and explains the difficulties when reading MotionBuilder's Python documents: *"Basically, it tells me nothing. It's really difficult to find out how to access the data I want through a python script. Here's almost no textual explanation. The name of variables is the only information I can get. Guessing and testing cost me lots of time."* Thus, the analyses of API documentation can significantly guide developers with better API learning experience.

Dataset limitation. The API crowd documentation is derived from the online data dump published on Stack Exchange. Despite that the data dump in our study is collected from January 2012 to June 2017, we also achieve high API coverage in the results of RQ1. Thus, we believe that the analyses between the API crowd documentation and official API tutorials can also get a convincing comparison.

6.2 Internal validity

Threats to internal validity are the quality of API crowd documentation and the analysis bias.

Quality of API crowd documentation. Our empirical study is based on the analysis between official API tutorials and API crowd documentation. The quality of the API crowd documentation may be a threat. In particular, if the API crowd documentation is composed of all the questions and the answers in our data dump, the APIs in lower quality of questions or answers will be calculated, which may confuse our analysis. Therefore, only the questions and the corresponding best answers are included in our study.

Analysis bias. When an API is simple enough, there need not abundant interpretations to introduce how to use the API. However, if the API is practical in most programming tasks, the frequency of the API in the official API tutorials would not as high as that in API crowd documentation, such as "java.lang.Double", "java.lang.Long", and "java.lang.Byte". This phenomenon may influence the analysis of the mismatch on the frequently used APIs between official API tutorials and API crowd documentation. However, the mismatch is universal in the comparison for most APIs, especially for the fine-grained APIs.

7 Conclusion

To the best of our knowledge, this paper is the first time to analyze the differences on the API learnability between official API tutorials and API crowd documentation. We conduct an empirical study based on three major aspects: (1) the coverage of APIs, (2) the concerns for APIs, and (3) the presentations. The study is performed on the Java API documentation and the Android API documentation.

Overall, we find that the coverage of APIs in API crowd documentation is higher than that in official API tutorials. Most APIs covered by official API tutorials are also covered by API crowd documentation. Thus, for most APIs, developers can refer to API crowd documentation to learn how to use them. APIs provided in official API tutorials might not focus on the APIs that developers need, which may decrease the API learnability for developers to some extent. Besides, official API

tutorials are helpful for developers to locate the API information due to the longer page. API crowd documentation can provide more quantitative and longer code snippets to help developers accomplish development tasks. These findings can help developers understand what API information is available in different API documents and find the useful information they need.

References

[1] Subramanian S, Inozemtseva L, Holmes R. Live API documentation. In *Proc. of the 36th Int. Conf. on Softw. Eng.*, May 2014, pp.643-652.

[2] Petrosyan G, Robillard M P, De Mori R. Discovering information explaining API types using text classification. In *Proc. of the 37th Int. Conf. on Softw. Eng.*, May 2015, pp.869-879.

[3] Maalej W, Robillard M P. Patterns of knowledge in API reference documentation. *IEEE Trans. on Softw. Eng.*, 2013, 39(9): 1264-1282.

[4] Robillard M P. What makes apis hard to learn? Answers from developers. *IEEE Softw.*, 2009, 26(6): 27-34.

[5] Thayer K. Using Program Analysis to Improve API Learnability. In *Proc. of the 2018 ACM Conf. on Int. Computing Education Research*, Aug. 2018, pp.292-293.

[6] Jiang J, Koskinen J, Ruokonen A, Systa T. Constructing usage scenarios for API redocumentation. In *Proc. of the 15th IEEE International Conference on Program Comprehension*, Jun. 2007, pp.259-264.

[7] Jiang H, Zhang J, Li X, Ren Z, Lo D. A more accurate model for finding tutorial segments explaining APIs. In *Proc. of the 2016 IEEE 23rd Int. Conf. on Softw. Analysis, Evolution, and Reengineering*, Mar. 2016, pp. 157-167.

[8] Zhang J, Jiang H, Ren Z, Chen X. Recommending APIs for API related questions in stack overflow. *IEEE Access*, 2018, 6: 6205-6219.

[9] Treude C, Storey M A. Effective communication of software development knowledge through community portals. In *Proc. of the 19th ACM SIGSOFT symposium and the 13th European Conf. on Foundations of Softw. Eng.*, Sep. 2011, pp.91-101.

[10] Robillard M P, Deline R. A field study of api learning obstacles. *Empir. Softw. Eng.*, 2011, 16(6): 703-732.

[11] Scaffidi C. Why are apis difficult to learn and use. *ACM Crossroads Student Magazine*, 2006, 12(4): 4-10.

[12] Jiang H, Zhang J, Ren Z, Zhang T. An unsupervised approach for discovering relevant tutorial fragments for APIs. In *Proc. of the 39th Int. Conf. on Softw. Eng.*, May 2017, pp.38-48.

[13] Ye X, Shen H, Ma X, Bunescu R, Liu C. From word embeddings to document similarities for improved information retrieval in software engineering. In *Proc. of the 38th Int. Conf. on Softw. Eng.*, May 2016, pp.404-415.

[14] Uddin G, Robillard M P. How API documentation fails, *IEEE Softw.*, 2015, 32(4): 68-75.

[15] Zhou Y, Gu R, Chen T, Huang Z, Panichella S, Gall H. Analyzing APIs documentation and code to detect directive defects. In *Proc. of the 39th Int. Conf. on Softw. Eng.*, May 2017, pp.7-37.

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

[17] Wang X, Huang C, Yao L, Benatallah B, Dong M. A survey on expert recommendation in community question answering. *Journal of Computer Science and Technology*, 2018, 33(4): 625-653.

[18] Mamykina L, Manoim B, Mittal M, Hripcsak G, Hartmann B. Design lessons from the fastest q&a site in the west. In *Proc. of the 2011 Annual Conf. on Human Factors in Computing Systems*, Apr. 2011, pp.2857-2866.

[19] Beyer S, Macho C, Pinzger M. On android API classes and their references on stack overflow. Technical Report, University of Klagenfurt, 2016. https://serg.aau.at/pub/StefanieBeyer/Publications/ techreport_AAU-SERG-2016-001.pdf

[20] Zhang Y, Lo D, Xia X, Sun J L. Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Tech.*, 2015, 30(5): 981-997.

[21] Yang X L, Lo D, Xia X, Wan Z Y, Sun J L. What security questions do developers ask? A large-scale study of stack overflow posts. *Journal of Computer Science and Tech.*, 2016, 31(5): 910-924.

[22] Rosen C, Shihab E. What are mobile developers asking about? A large scale study using stack overflow. *Empirical Softw. Eng.*, 2016, 21(3): 1192-1223.

[23] Barua A, Thomas S W, Hassan A E. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Softw. Eng.*, 2014, 19(3): 619-654.

[24] Chen C, Wu K, Srinivasan V, Bharadwaj R K. The best answers? Think twice: identifying commercial campagins in the CQA forums. *Journal of Computer Science and Tech.*, 2015, 30(4): 810-828.

[25] Brito G, Hora A C, Valente M T, Romain R. On the use of replacement messages in API depreca-tion: An empirical study. *Journal Syst. Softw.*, 2018, 137: 306-321.

[26] Dagenais B, Robillard M P. Recovering traceability links between an API and its learning resources. In *Proc. of the 34th Int. Conf. on Softw. Eng.*, June 2012, pp.47-57.

[27] Rastkar S, Murphy G C, Murray G. Summarizing software artifacts: a case study of bug reports. In *Proc. of the 32nd Int. Conf. on Softw. Eng.*, May 2010, pp.505-514.

[28] Bettenburg N, Just S, Schröter A, Weiss C, Premraj R, Zimmermann T. What makes a good bug report? In *Proc. of the 16th ACM SIGSOFT Int. Symposium on Foundations of Softw. Eng.*, Mar. 2008, pp. 308-318.

[29] Schneider T D. Information theory primer with an appendix on logarithms. *National Cancer Institute*, 2007.

[30] Smith E A, Senter R J. Automated readability index. *Aerospace Medical Research Laboratories*, 1967, pp.1-14.

[31] Jay G T, Hale J E, Smith R K, Hale D P, Kraft N A, Ward C. Cyclomatic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship. *Journal of Softw. Eng. and Applications*, 2009, 02(03): 137-143.

[32] Nykaza J, Messinger R, Boehme F, Norman CL, Mace M, Gordon M. What programmers really want: results of a needs assessment for SDK documentation, In *Proc. of the 20th annual ACM SIGDOC int conf. on computer documentation*, Oct. 2002, pp.133-141

[33] Mclellan S G, Roesler A W, Tempest J T, Spinuzzi C I. Building more usable APIs. *IEEE Softw.*, 1998, 15(3): 78-86.

[34] Santos A L, Myers B A. Design annotations to improve API discoverability. *Journal of Systems and Softw.*, 2017: 17-33.

[35] McCabe T J. A complexity measure. *IEEE Trans. on Softw. Eng.*, 1976, 4: 308-320.

[36] Yuan T, Thung F, Sharma A, Lo D. APIBot: question answering bot for api documentation. In *Proc. of the 32nd IEEE/ACM Int. Conf. on Automated Software Engineering*, Oct. 2017, pp.153-158.

[37] Zar J H. Spearman rank correlation. *Encyclopedia of biostatistics*, 1998, 5:4191-6.

[38] Mandelin D, Xu L, Bodik R, Kimelman D. Jungloid mining: helping to navigate the API jungle. In *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2005, 40(6): 48-61.

[39] Mann H B, Whitney D R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 1947, 18(1): 50-60.

[40] Kitchenham B A, Pfleeger S L. Personal opinion surveys. In *Guide to Advanced Empir. Softw. Eng.*, 2008, pp.63-92.

[41] Zou W, Lo D, Chen Z, Xia X, Feng Y, Xu B. How practitioners perceive automated bug report management techniques. *IEEE Trans. on Softw. Eng.*, 2018.

[42] David L O, Nagappan N, Zimmermann T. How practitioners perceive the relevance of software engineering research. In *Proc. of the 10th Joint Meeting on Foundations of Soft. Eng.*, 2015, pp.415-425.

[43] Fisher R A. On the interpretation of x2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, 1922, 85(1):87-94.

[44] McDonald J H. Handbook of biological statistics. *Sparky House*, 2009.

[45] Rocha A M, Maia M A. Automated API documentation with tutorials generated from stack overflow. In *Proc. of the 30th Brazilian Symposium on Softw. Eng.*, Sep. 2016, pp.33-42.

[46] Kim J, Lee S, Hwang S, Kim S. Enriching Documents with Examples: A Corpus Mining Approach. *ACM Transactions on Information Systems*, 2013, 31(1): 1-27.

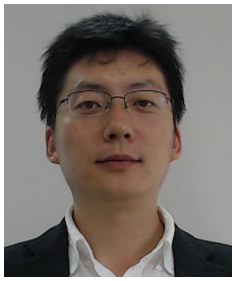
[47] Treude C, Robillard M P. Augmenting API documentation with insights from stack overflow[C]. In *Proc. of the 38th Int. Conf. on Softw. Eng.*, May 2016, pp.392-403.

Yi-Xuan Tang received the B.S. degree in computer science and technology from Liaoning University, Shenyang, in 2015. She is currently a Ph.D. candidate in Dalian University of Technology, Dalian. Her current research interests include software data analytics and compiler testing.

Zhi-Lei Ren received the B.S. degree in software engineering and the Ph.D. degree in computational mathematics from Dalian University of Technology, Dalian, in 2007 and 2013, respectively. He is currently a professor at Dalian University of Technology, Daian. He is a member of the China Computer Federation (CCF) and the Association for Computing Machinery (ACM), respectively. His current research interests include evolutionary computation, automatic algorithm configuration, and mining software repositories.

He Jiang is currently a professor with Dalian University of Technology and an adjunct professor with Beijing Institute of Technology. He is a member of the China Computer Federation (CCF), the Association for Computing Machinery (ACM), and the Institute of Electrical and Electronics Engineers (IEEE), respectively. His current research interests include search-based software engineering and mining software repositories. He has published over 60 referred papers on journals and international conferences, including IEEE Trans. Software Engineering, IEEE Trans. Knowledge and Data Engineering, ICSE, SANER, etc., supported by the Program for New Century Excellent Talents in University and the National Science Fund for Excellent Young Scholars. In addition, he serves as the guest editors of some journals and magazines, including IEEE Computational Intelligence, Journal of Computer Science and Technology, Frontiers of Computer Science, etc.

Xiao-Chen Li is currently a research associate at University of Luxembourg. He received the Ph.D. degree in software engineering from Dalian University of Technology, Dalian, China, in 2019. He is a member of the China Computer Federation (CCF). His current research interests are mining software repositories (MSR) and software semantic analysis. He has published referred papers on premier journals and international conferences, including TSE, TRE, ICSE and ICPC. More information about him is available online at https://xiaochen-li.github.io.



Wei-Qiang Kong received the Ph.D. degree in information science from Japan Advanced Institute of Science and Technology, in 2006. He is a member of the China Computer Federation (CCF). He is currently a professor at Dalian University of Technology, Dalian. His current research interests include software engineering and formal methods (formal verification).