2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)

Towards Crowd-Sourced API Documentation

Gias Uddin
SWAT Lab
Polytechnique Montréal, QC, Canada
gias.uddin@mail.mcgill.ca

Foutse Khomh
SWAT Lab
Polytechnique Montréal, QC, Canada
foutse.khomh@polymtl.ca

Chanchal K Roy
Department of Computer Science
University of Saskatchewan, SK, Canada
chanchal.roy@usask.ca

***Abstract*—The learnability of an API suffers when the official documentation of the API is incomplete. Developers discuss usage scenarios of APIs in the online developer forums. As such, by automatically mining such crowd-sourced documentation of APIs, we can address the shortcomings of API official documentation. We present a framework to automatically mine usage scenarios about APIs from online developer forums. Each usage scenario of an API consists of a code example, a summary description, and the reactions (i.e., positive and negative opinions) of other developers towards the code example. We evaluate our API usage mining framework by producing a benchmark dataset. We observed a precision of 0.947 and a recall of 1.0 with the linking of a code example to an API mention in the forum.**

***Index Terms*—API, Mining, Usage, Documentation.**

I. Introduction

APIs (Application Programming Interfaces) offer interfaces to reusable software components. The learnability of an API depends on the availability of learning resources of the API [1]. Unfortunately, despite developers' reliance on API official documentation as a major resource for learning and using APIs, the documentation can often be incomplete [2].

The shortcomings in the official documentation led to the creation and popularity of online developer forums (e.g., Stack Overflow), where developers can ask and answer questions on how to address diverse development tasks that may involve APIs. A number of research efforts have focused on integrating information from the forum posts into API official documentation, such as the linking of API types in Javadocs to code examples in forum posts [3], the presentation of interesting textual contents from Stack Overflow about an API type in Javadocs [4] and so on. Unfortunately, the approaches can have the following shortcomings: 1) The traceability techniques cannot link a code example to an API mentioned in the post texts and about which code example is *actually* provided, 2) The techniques do not provide any textual description of what the code example does, and 3) The techniques do not offer any insights into the quality of the mined code examples.

We present a framework to automatically mine *API usage scenarios* from developer forum posts. Each scenario consists of four items: **(1)** A code example as discussed in a forum post, **(2)** An API about which the code example is provided to address a development task, **(3)** A natural language description of the code example to summarize what the code does, and **(4)** A list of reactions as positive or negative opinions from other developers towards the code example.

Our proposed API usage scenario mining framework works as follows. First, given as input a forum post that consists of a code example, we automatically link the code example to an API mentioned in the textual contents of the forum post (i.e., the code example is provided to discuss a use case of the API). Second, given as input a code example in a forum post that is linked to an API mention, we associate a summary description of the textual contents of the post where the code example is found. Third, we associate the positive and negative opinionated sentences as reactions towards the code example. In a benchmark-based study of 300 randomly sampled code examples from Stack Overflow, we observed that our algorithm to link a code example to an API mention in forum post showed a precision of 0.947 and a recall of 1.0. We compared the algorithm against two state of the art baselines. Our algorithm outperformed all the baselines.

II. Related Work

The focus of this paper is to establish a framework to automatically mine usage scenarios about APIs from developer forums that can facilitate task-orientation documentation for APIs. We follow the concept of "minimal manual" as pioneered by Carroll et al. [5] which promotes task-centric documentation of manual and is proven to work better than the traditional API documentation. We differ from the above work as follows: 1) We include comments as posted in a forum post as reactions to a code example in our usage scenarios. 2) We automatically mine such usage scenarios from online forum posts, thereby greatly reducing the time and complexity that may be required to produce those manually. Our algorithm to mine usage scenarios differ from the state of the art techniques to link code examples to APIs in forum posts [3], [6], because we detect an API as an API name as mentioned in the textual contents in the forum posts. Unlike the above work that consider each class of a software library as an API, we consider the entire library as an API. This design decision is based on our observation of how API names are actually mentioned and discussed in the forum posts. In contrast to Subramanian et al. [3] that needs the construction of an offline API database, our technique can rely on online API databases. Unlike [6], we do not rely on the analysis of client software code to infer usage patterns of an API. While such analysis can offer better accuracy than Subramanian et al. [3] using API databases, the approach is not feasible when such client software code may not be available (e.g., for a new API).

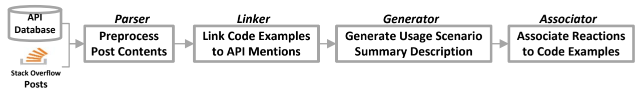
API Database
Stack Overflow Posts
Parser
Preprocess Post Contents
Linker
Link Code Examples to API Mentions
Generator
Generate Usage Scenario Summary Description
Associator
Associate Reactions to Code Examples

Fig. 1. The major components of our API usage scenario mining framework

III. The Framework

Our mining framework takes as input a forum post and outputs all the usage scenarios found in the post. Our framework consists of five major components (Figure 1):

C1. An **API database** to identify the API mentions. Our API database consists of open source and official Java APIs. An API consists of one or more modules. Each module can have one or more code packages. Each package can have one or more code elements, such as classes, methods, and so on.

C2. A suite of **Parsers** to preprocess the forum post contents. Given as input a forum post, we preprocess its content as follows: (1) We categorize the post content into three types: (a) *code snippets*; (b) *hyperlinks*; and (c) *natural language text* representing the rest of the content. (2) We detect individual sentences in the *natural language text*. (3) Following Dagenais and Robillard [7], we discard the following *invalid* code examples during our parsing: (a) Non-code snippets (e.g., XML, JSON extract), (b) Non-Java snippets (e.g., JavaScript). The rest of the code examples are considered as *valid*.

C3. A **Linker** to associate a code example to an API mention. Given as input a code example in a forum post, we associate it to an API mentioned in the post in two steps: a) We detect API mentions in the textual contents of forum posts following [8]. Therefore, each API mention is a token (or a series of tokens) if it matches at least one API or module name. b) We associate a code example in a forum post to an API mention by learning how API elements in the code example may be connected to a candidate API in the mention candidate lists of the API mentions in the same post. For example, if we have two API mentions in texts as 'Jackson' and 'Gson' and the code example has API elements from Jackson an java.util APIs, we associate the code example to the mention 'Jackson'.

C4. A **Generator** to produce a natural language summary description of a code example. Our algorithm to generate natural language summary description of a code example takes as input all the textual contents of Stack Overflow thread where the code example is found and outputs a short textual summary description of the code example. Our algorithm works by first finding sentences where the API associated to a code example is mentioned both *explicitly* (e.g., by the API name) and *implicitly* (e.g., by a reference, such as a pronoun). We produce summary description only for code examples that are found in the answers to questions. This is based on the observation that such a code example is in more need to be understood within the context of a development task [3].

C5. An **Associator** to find reactions towards code examples. The inputs to the algorithm are all the comments towards the post where the code example is found. The output is a list of opinionated sentences that are related to the code example.

IV. Evaluation

The effectiveness of a crowd-sourced API documentation technique relies on the correct linking of a code example to an API about which the code example is provided. To assess the accuracy of our algorithm to link code examples to API mentions, we produced a benchmark that contained randomly selected 300 code examples from Stack Overflow threads tagged as 'Java', each linked to an API mention after manual assessment. We report using three performance measures: precision (P), recall (R), and F-measure ($F1$).

$$P = \frac{TP}{TP+FP},\ R = \frac{TP}{TP+FN},\ F1 = 2 * \frac{P*R}{P+R},$$

TP = Nb. of true positives, and FN = Nb. false negatives.

We compare our algorithm against two baselines: 1) Baker [3], and 2) Google Search engine. We achieved a precision of 0.947 and a recall of 1.0 using our algorithm. The high recall was due to the greedy approach of our algorithm, i.e., the algorithm always assigns each code example to an API. Almost one-third of the wrong associations happened due to the the code example being too short to provide enough contexts for linking. The baseline Baker shows the best precision among all (0.966), but with the lowest recall (0.473). This is because most of the code examples included multiple APIs, but the post texts actually discussed only one of those APIs. The Google search results show the lowest precision (0.388), confirming the assumption that Google is primarily a generally purpose search engine.

V. Conclusions

Developers discuss API usage scenarios in forum posts. We present a framework to automatically mine API usage scenarios from forums. To assist developers in their development task completion using the mined scenarios, we developed an online search and summarization engine for API usage scenarios. Our future work will focus on the understanding of the effectiveness of the framework to support API documentation.

References

[1] M. P. Robillard and R. DeLine, "A field study of API learning obstacles," *Empirical Software Engineering*, vol. 16, no. 6, pp. 703–732, 2011.
[2] G. Uddin and M. P. Robillard, "How API documentation fails," *IEEE Softawre*, vol. 32, no. 4, pp. 76–83, 2015.
[3] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live API documentation," in *Proc. 36th International Conference on Software Engineering*, 2014, p. 10.
[4] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from stack overflow," in *Proc. 38th International Conference on Software Engineering*, 2016, pp. 392–403.
[5] J. M. Carroll, P. L. Smith-Kerker, J. R. Ford, and S. A. Mazur-Rimetz, "The minimal manual," *Journal of Human-Computer Interaction*, vol. 3, no. 2, pp. 123–153, 1987.
[6] H. Phan, H. A. Nguyen, N. M. Tran, L. H. Truong, A. T. Nguyen, and T. N. Nguyen, "Statistical learning of api fully qualified names in code snippets of online forums," in *Proceedings of 40th International Conference on Software Engineering*, 2018, pp. 632–642.
[7] 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.
[8] G. Uddin and M. P. Robillard, "Automatic resolution of API mentions in informal documents," in *McGill Technical Report*, 2017, p. 6.