CMPUT 499: Mining Sofware Repositories

Literature Review

Monica Bui

1 Introduction

Third-party software libraries and Application Programming Interfaces (APIs) offer a way for developers to use existing features and funuctionalities to build their projects without having to re-invent the wheel. There are many libraries to use for different programming languages whether it is open source or hosted elsewhere. Although having this much freedom for deciding on what libraries to use is flexible, it is also conflicting to figure out which one is best for your project. It is difficult to choose because there are a variety of factors involved such as functionality and developer support. In this review, we look at several diferent articles to help analyze various properties of libraries and to determine a high quality method of comparison between them.

2 Article Review

2.1 Library Recommenders

One problem is researching new analogical libraries to use for different programming languages that has similar functionalities to the ones you currently know. Chen's et. al [1] paper discusses their library recommender that compiles a list of libraries from community resources such as blogs and Q&A sites like StackOverflow [2] and outputs a list of recommended libraries in the developer's language of choice.

This is implemented by first mining tags on questions posted online on StackOverflow [2]. These tags are split into two knowledge bases: relational and categorical. Respectively, relational knowledge is how pairs of tags are correlated to each other eg. Java and JUnit while categorical knowledge consists of how tags are grouped into categories such as language, operating system, concept, or library with both bases being analyzed by NLP. The key idea here is that having different seperated tag categories and relatonships between tags often mentioned together allowed for a simpler way to recommend a new library. With the database in place, users can then search for recomendations through the built web application called SimilarTech [3].

While trying out the application myself, I found that search results would only yield for libraries that are mentioned in specific contexts. Axios [4] a popular Javascript HTTP library should output expected recomendations

like Requests [5] module for Python but the actual list printed was empty. Axios [4] itself is mentioned around many situations like REST, APIs, and HTTP requests making it difficult to figure out in what context it should be recommended.

While the number of languages it can suggest libraries for is limited to 5, the precision metric is impressive with 1 language at 81% and with 5 being at 67% showing its potential to grow in the future.

Going back to the key problem of deciding on the right library to use, Uddin's et. al [6] article highlights their approach to this by looking at personal developer opinions's on different resources and how it affects the reader's decision. Furthermore, the sentiment behind this can be used to indicate if its a positive, questionable, or a negative API to use. The tool created, Opiner [7] is a summarization engine that examines these opinions and evaluates its sentiment to see if the API should be recommended also displaying ratings on API traits and organizing top opinions on both positive and negative sides.

The backend of the application web crawls through a Q&A site, StackOverflow [2] to extract answer information surrounding an API topic. This data is used to help discover when an API in mentioned, what kinds of opinion are associated with an API, and combining these opinions under common aspects developers care about such as usability and performance.

Using Opiner [7] through a small research study evaluated on 2 seperate groups of particpant's choices to see if just StackOverflow [2] alone would work for decision making on selecting an API or both StackOverflow and Opiner would be better. We view that devlopers are more confident in their selection with both tools vs just StackOverflow alone.

A weakness is that although the evaluation of the study proves useful, having 100% as a metric for using both StackOverflow and Opiner would be hard to generalize for a larger population.

2.2 Github Badges

Many developers like to both use and contribute to open-source software projects on GitHub [8]. With many online software to use, it's difficult to pin point which is worth your time to contribute to and integrate with your project. Trockman et. al [9] looks at the in depth quality of a package by

Figure 1: Example badges in a open source PHP repository [10]

analyzing repository badges eg. Figure 1 that maintainers display on their README.

Trockman not only mentions that badges are signals to make qualities of a project more transparent but also "may impact users' and contributers' decision making" [9]. Game like elements known as *Gamification* are not explicitly embedded into these badges but in reality motivate users to contribute higher quality code to increase the signal quality displayed by these visuals. The key questions to ask are, *What are the most common badges and what does displayong them intend to signal?* and *To what degree do badges correlate with qualities that developers expect?* [9]. To examine the impact of badges, Node Package Modules (npm) [11] is used as the research repository as it's the largest online collection of Javascipt packages.

Data is collected by mining all npm [11] packages and keeping those with metadata that included important metrics and a public Github [8] repository. They extract the badges through the git history associated with the README file by matching the regular expression for a badge insertion then further categorizing them into specific categories such as quality assuracne, popularity, dependencies etc. which each have different signaling intentions. With their survey insights, they develop sub questions to validate the qualities the badges are supposed to show. For each type of signal (dependencies, popularity, test suite, and quality pull requests), they look at impact before and after badge adoption through longitudinal analysis, statistical regressions to see how it is correlated with their arguments, and any underlying indicators that the badge may not overly express at first glance.

The results suggest that displaying badges highly correlate with better code practices specifically higher test coverage, updated dependencies, and higher quality code. However, overwhelming your repository with badges loses its intended signaling effect thus turns away users leading to decrease in downloads. Assessment signals that are more costly to produce are more

reliable than static conventional badges that display trivial information already readily found on the page.

With their survey design research method that collected important libraries metrics from contributers and maintainers however, having such a low response rate of 15.3% is difficult to generalize that these results will stay reliable for future studies.

2.3 Metrics

With software libaries and APIs evolving at a rapid speed, upgrading to the latest software can be cumbersome for developers as challenges of backwards incompatability and deciding on new APIs for your project prove to be a problem. Hora et. al [12] dicusses these 2 obstacles and implements a web application *apiwave* [13] to mitigate these problems and highlight API popularity and migration in depth with the former measured by the number of users using the service.

The research method takes the Git history code of a repository and outputs information about the API's popularity metric, method of migration with code snippets as examples. This code is further extracted by its Git diff. Based on the insertions and deletions, we can detect which lines have been modified for migration and update the popularity statistic by 1 suggesting that the old library lost a follower and the new library gained one with further assistance of mining import statements to see which APIs have changed. The client side of *apiwave* [13] can present a library at many levels including specific interface lookup eg. java.util vs java.util.Map. *Apiwave* also displays addition and deletions statistics, overall popularity graph, and code snippets that highlights recommended libraries to transfer over to based on diffs.

These results indicate different popularity trends and has been able to answer real life StackOverflow [2] questions regarding API migration. This helps to recommend the best software to use for developers based on ease of use through migration and the number of others who support this – popularity.

Even though there are a high number of vistors on the page, it is only limited to Java at the moment even though there are many lanugages out there such as Javascript and Python that host many open source libaries.

TODO) Talk about Fernando's article

3 Conclusion

For future software metric analysis research based on popular metrics found by Mora et. al [14] and Trockman et al. [9], we hope to create high quality, assessment Github [8] badges based on the most popular software metrics to help recommend the best libraries for users to use through first glance at the README file.

References

[1] C. Chen, S. Gao, and Z. Xing, "Mining analogical libraries in q amp;a discussions – incorporating relational and categorical knowledge into word embedding," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, March 2016, pp. 338–348.

[2] Stack-Oveflow, "Stack overflow," http://www.stackoverflow.com.

[3] Similartech, "Similartech," https://graphofknowledge.appspot.com/similartech.

[4] Axios, "Axios," https://github.com/axios/axios.

[5] Requests, "Requests," https://github.com/requests/requests.

[6] G. Uddin and F. Khomh, "Opiner: An opinion search and summarization engine for apis," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Oct 2017, pp. 978–983.

[7] Opiner, "Opiner," http://sentimin.soccerlab.polymtl.ca:38080/opinereval/.

[8] Github, "Github," "http://github.com/".

[9] A. Trockman, S. Zhou, C. Kästner, and B. Vasilescu, "Adding sparkle to social coding: An empirical study of repository badges in the npm ecosystem," in *International Conference on Software Engineering*, ser. ICSE. ACM, 2018, pp. 511–522.

[10] Marabesi, "Github badges!(php repository)," 2015, [Online; accessed Sept 19, 2018]. [Online]. Available: https://marabesi.com/wp-content/uploads/2015/06/badges2.png

[11] NPM, "Npm js," "https://www.npmjs.com/".

[12] A. C. Hora and M. T. Valente, "Apiwave: Keeping track of api popularity and migration." in *ICSME*. IEEE Computer Society, 2015, pp. 321–323.

[13] apiwave, "apiwave," "http://apiwave.com/".

[14] F. L. de la Mora and S. Nadi, "Which library should I use?: a metric-based comparison of software libraries," in *ICSE (NIER)*. ACM, 2018, pp. 37–40.