tool-recommender-bot

Chris Brown and Emerson Murphy-Hill
Department of Computer Science
North Carolina State University
Raleigh, NC
Email: dcbrow10@ncsu.edu, emerson@csc.ncsu.edu

***Abstract*—Recommendation systems were developed to improve the adoption of useful software tools and features designed to save time and effort in completing tasks that are often ignored by users. Previous research suggests that peer-to-peer recommendations are the most effective mode of tool discovery and that the receptiveness of recommendees is the most important characteristic in determining the outcome of tool recommendations. To help increase awareness of useful tools, we developed and evaluated a new recommendation system tool-recommender-bot designed to integrate aspects of peer interactions and user receptivity into automated tool suggestions for software developers of real-world applications. Our findings suggest that tool-recommender-bot is awesome, cool, and very effective in improving tool discovery.**

***Index Terms*—Software Engineering; Tool Recommendation; Tool Discovery; Open Source**

I. Introduction

Tool discovery is a problem...

Automated recommendation systems can help solve this problem...

But existing recommendations systems are ineffective...

Peer interactions and receptiveness are effective [1]...

We created tool-recommender-bot to solve this...

RQ1: How often can we expect tool-recommender-bot to make recommendations?
RQ2: How useful are recommendations from tool-recommender-bot to developers?

To answer these questions, we conducted a study analyzing tool-recommender-bot on five? popular open source Java projects to observe how many tool suggestions would be made based on past changes to the code base and how software developers reacted to receiving recommendations. This research makes the following contributions:

- We introduce the design and implementation of a novel automated recommendation system tool-recommender-bot
- We provide implications for future

II. Related Work

Improving tool discovery...

Existing automated tool recommendation systems...

III. Tool

tool-recommender-bot is awesome. Here's how...

A. Implementation

tool-recommender-bot technical details...

1) Jenkins:

2) Error Prone:

B. Receptiveness

tool-recommender-bot was designed to integrate characteristics of peer interactions into automated recommendations. To better understand user-to-user recommendations and why they are effective for tool discovery, we observed how colleagues recommend tools to each other while completing tasks in prior work. Our results found that the receptiveness of users was the only significant indicator of determining whether a tool recommendation was effective or not compared to other interaction characteristics, such as politeness [1]. Fogg defines receptiveness using two criteria, *demonstrating a desire* and *familiarity* with the adopted behavior [3].

a) Desire:

b) Familiarity:

IV. Methodology

A. Projects

To evaluate the effectiveness of our recommendation system, we implemented tool-recommender-bot on five real-world open-source software applications. We selected projects hosted on Github[1], a popular source code management site that hosts thousands of code repositories online. To narrow down projects for our evaluation, we picked Github projects that met the following criteria:

- one of the top Trending projects[2] on GitHub based on activity by the community at the time of this writing,
- primarily written in the Java programming language[3],
- owned by a Github organization instead of personal user accounts,
- and used Maven[4] for software build management.

[1]https://github.com
[2]https://github.com/trending
[3]https://java.com
[4]https://maven.apache.org/

TABLE I
EVALUATION PROJECTS

Project	Java Files	LOC	Pull Requests
spring-boot[5]			
dubbo[6]			
guava[7]			
retrofit[8]			
rocketmq[9]			

Details on projects used for study including GitHub repository name, number of Java files, lines of Java code, and total pull requests.

Table I presents the projects used for evaluating tool-recommender-bot in our study and provides details about each repository. These repositories span a wide range of organizations and real-world software applications, including a system to create "stand-alone, production-grade" Spring applications [6], a high-performance remote procedure call framework from Chinese e-commerce company Alibaba [2], a collection of core Google libraries [4], an HTTP client for Android financial services mobile application Square [7], and a distributed messaging and data-streaming platform by Apache [5].

B. Study Design

We divided our study into two segments to address each research question:

1) RQ1: Last 100 pull requests on repositories...

2) RQ2: Followed up with pull request authors to gather data on recommendation...

V. Results

A. How often can we expect tool-recommender-bot to make recommendations?

Tons of recommendations...

No false positives...

B. How useful are recommendations from tool-recommender-bot to developers?

Excellent responses from recommendees...

Statistically significant data...

VI. Discussion

A. Observations

B. Implications

Here's what our results say about ways to improve tool recommendation systems...

VII. Limitations

Internal

An external threat to the validity of our study is that we only observed open source projects hosted on Github in our evaluation. Our results may not generalize to closed source software projects and their developers. To minimize this, we selected popular real-world software applications on Github owned by organizations to avoid the use of personal development projects. Additionally, our recommendation system has limited generalizablility due to the fact we currently only assess recommendations for the Error Prone static analysis tool on Java projects that build with Maven. Future work will look to extend tool-recommender-bot to include different types of tools, programming languages, and build systems.

VIII. Future Work

More tools to recommend (static analysis, security, etc.)

More programming languages instead of just java...

More build systems (ant, gradle, TravisCI, bazel)...

IX. Conclusion

tool-recommender-bot is awesome

References

[1] C. Brown, J. Middleton, E. Sharma, and E. Murphy-Hill. How software users recommend tools to each other. In *Visual Languages and Human-Centric Computing*, 2017.

[2] Dubbo. https://github.com/alibaba/dubbo.

[3] B. Fogg. Creating persuasive technologies: An eight-step design process. In *Proceedings of the 4th International Conference on Persuasive Technology*, Persuasive '09, pages 44:1–44:6, New York, NY, USA, 2009. ACM.

[4] Google guava. https://github.com/google/guava/wiki.

[5] Apache rocketmq. https://rocketmq.apache.org/.

[6] Spring boot. https://projects.spring.io/spring-boot/.

[7] Square retrofit. http://square.github.io/retrofit/.