Poster: Forks Insight: Providing an Overview of GitHub Forks

Luyao Ren
Peking University

Shurui Zhou
Carnegie Mellon University

Christian Kästner
Carnegie Mellon University

Andrzej Wąsowski
IT University of Copenhagen

1 INTRODUCTION

Fork-based development allows developers to start development from existing software repository by copying the code files, and gives developers the freedom and independence to make modifications on their own fork [2, 3, 5, 10].Fork-based development has been widely used in both open source communities and industry. But when the number of forks grows, it becomes difficult for programmers to keep track of decentralized development activity in many forks. Some of the developers on Github we have interviewed previously reported the problems they faced in terms of losing overview of forks. For example, one said: *"I do not have much visibility of the forks. They are too many and it is overwhelming to keep track of them"* [11]. Both Duc et al. and Berger et al. (Luyao: still confused with CK marked here.)found that this problem also appears in industrial that it's hard for individual teams to know who is doing what and what code changes are made in other forks [1, 4].

Even though GitHub supports a network view, it is difficult to gain a straightforward overview of specific activities in forks. As another developer said: *"The network view is helpful for seeing how active a fork is, but often you have to scroll back a lot to find the fork point and then you have to go to the end again for seeing what changed since then in the parent and in the fork, by reading the tooltips of each commit."*[1] A lack of an overview of forks leads to several additional problems: (1) redundant development: developers may re-implement functionality has already been developed elsewhere; (2) lost contributions: the contributions developers made are easily lost to the larger community unless they contribute those changes back to the original project; (3) suboptimal forking point: developers might not fork from the codebase that is closest to their intended goals [3, 9, 11]. We therefore argue that (Luyao: use "argue" here?) it's necessary to give developers a panoramic view to help them better understand activities among various forks.

Forks Insight(http://www.forks-insight.com) is a more lightweight and accessible solution of our prior academic prototype INFOX [11]. INFOX analyses and clusters changes in C/C++ code within forks. While Forks Insight offers an web service for all GitHub repositories. It provides an overview of each repository in fork-level granularity and delivers insights to interested developers including repository's maintainers and other developers who are interested in the repositories.

2 FORKS INSIGHT

Forks Insight provides facilities to explore unintegrated changes to find opportunities for reuse, to find inspirations for further development, to connect developers working on similar topics. Forks Insight analyzes each active fork of a repository and takes the diff between the fork point and the latest commits (Shurui: need to check) of the fork and extracting keywords from code changes, comments and commit messages. Besides, Forks Insight presents statistical data of code changes in the granularity of commit, file and line. The user interface of Forks Insight is shown in Fig.1.

Users login with their own GitHub accounts, and subscribe repositories on GitHub by importing them from their public repositories or searching for a repository url. Forks Insight will start crawling and analyzing data if the repository is not in the database yet and remind the user when finished.

2.1 Extracting Keywords

To give developers a quick idea of what fork changed, we build on INFOX's idea of labeling features [11] to generate the column of representative keywords in Forks Insight.(Luyao: need to check) We use well-known natural language processing technique TF-IDF [8] on text related to code changes, such as source code, comments, and corresponding commit messages. First, we do some preprocessing: remove all the numeric strings; split for underline-separated and Camel-Case cases; lemmatize words to a normal form. Second, we use TF-IDF technique to extract keywords from the text. Though TF-IDF can effectively filter some stop words like "or", "and", there are still some words with high weight like "public", "private" which is common in language like Java/C++. To improve our result, we manually added the stop words list for different programming language corresponding to their different characteristics.

2.2 Tagging

Developers fork a repository for different reasons: adding new features, fixing bugs, and changing configuration, etc. [3, 6, 7, 9]. Since tagging is a simple and intuitive way to summarize the purpose of forking and also convenient to cluster similar forks. So we add a column at the end of each row to allow user to annotate the tags manually on forks. We hope user's contribution on tags can not only help themselves maintain and understand each fork which may reduce the redundant development, but also help the whole open source community, especially for the new users who are interested in this repository but not familiar with repositories. The data of tagging will also be useful for future research work.

(Shurui: Fork Insight allows users to tag each fork by the main activity of each fork based on their understanding. This data could help users to manage and classify forks with similar goals.)

2.3 Searching

The interview we did in INFOX shows the problem of redundant development do exist in forks. For example, a developer said :*"It does look like somebody did a very simple one-function [...] system.*

[1] https://github.com/dear-github/dear-github/issues/175

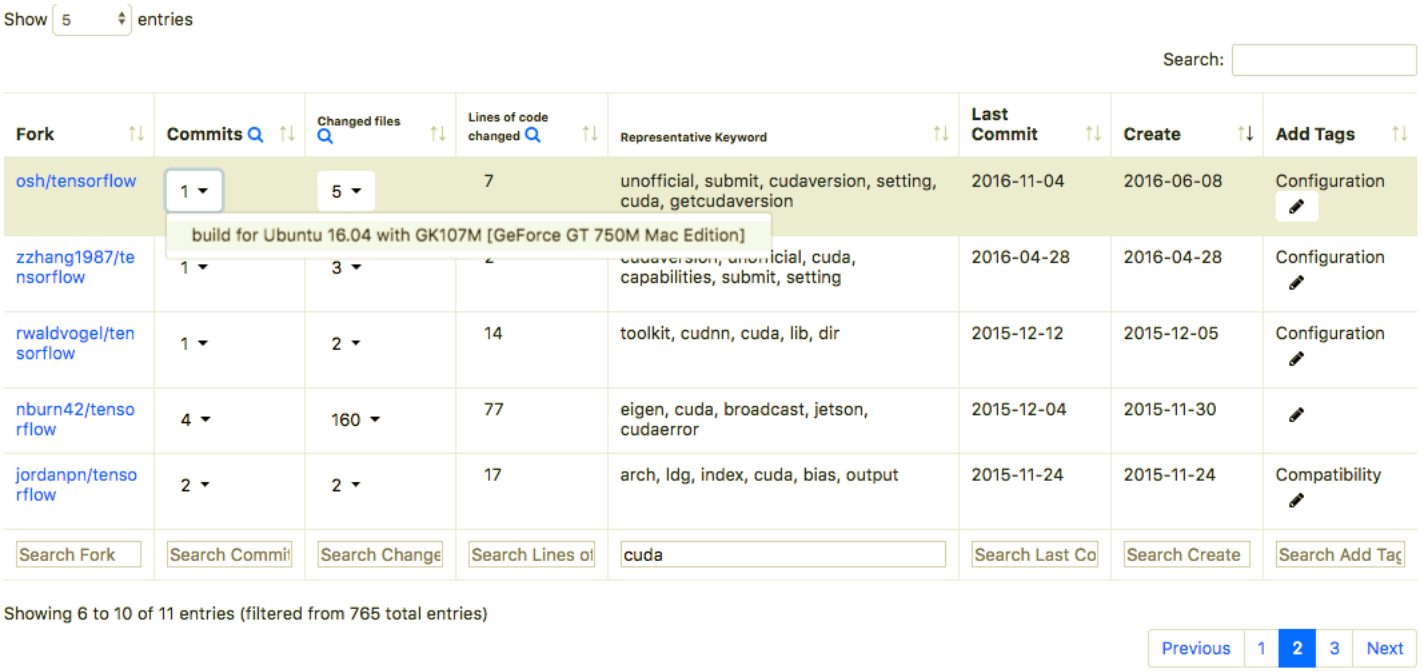

Figure 1: User Interface of Forks Insight.(Luyao: If this example is fine, I will add extra explaination.)

[checking the code] I think they should use our code, there is great reason to use it." Another one said: *"I can see multiple forks are working on the similar problem. This one looks like it is adding [...] that I already added"* [11].

Using searching keywords in Forks Insight is a simple and practical way to try to solve it. For example, in Fig. 1, searching for "cuda" in forks of tensorflow(an open-source software library developed by Google) will get several forks and most of them are related to GPU configuration. The example shows that by using keyword search in Forks Insight, user could find out some similar forks that could probably cut down the possibility of the redundant development.

3 CONCLUSIONS AND FUTURE WORK

We implemented Forks Insight to help developers get an overview of forks. The current release version focuses on simple analytics for the high level overview which is lightweight, scalable and practical. it uses the keyword extraction of INFOX and extends it with a user-friendly interactive web interface and features for searching and tagging. In order to improve the usability of Forks Insight, we plan to design a user study. And we would like to add more interactive elements and powerful functions into our tool. There are several directions we are considering to move forward: using more visualization to show the meaningful data; identifying features in forks; summarizing the activities of forks by natural language.

ACKNOWLEDGMENTS

TODO

REFERENCES

[1] Thorsten Berger, Divya Nair, Ralf Rublack, Joanne M Atlee, Krzysztof Czarnecki, and Andrzej Wąsowski. 2014. Three cases of feature-based variability modeling in industry. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 302–319.

[2] Jürgen Bitzer and Philipp JH Schröder. 2006. The impact of entry and competition by open source software on innovation activity. *The economics of open source software development* (2006), 219–245.

[3] Yael Dubinsky, Julia Rubin, Theodore Berger, Slawomir Duszynski, Matthias Becker, and Krzysztof Czarnecki. 2013. An exploratory study of cloning in industrial software product lines. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*. IEEE, 25–34.

[4] Anh Nguyen Duc, Audris Mockus, Randy Hackbarth, and John Palframan. 2014. Forking and Coordination in Multi-platform Development: A Case Study. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14)*. ACM, New York, NY, USA, Article 59, 10 pages.

[5] Neil A Ernst, Steve Easterbrook, and John MyLOPOULOS. 2010. Code forking in open-source software: a requirements perspective. *arXiv preprint arXiv:1004.2889* (2010).

[6] Tommi Mikkonen and Linus Nyman. 2011. To Fork or Not to Fork: Fork Motivations in SourceForge Projects. *Int. J. Open Source Softw. Process.* 3, 3 (July 2011), 1–9.

[7] Gregorio Robles and Jesús M. González-Barahona. 2012. A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes. In *Open Source Systems: Long-Term Sustainability - 8th IFIP WG 2.13 International Conference, OSS 2012, Hammamet, Tunisia, September 10-13, 2012. Proceedings*. 1–14.

[8] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[9] Ştefan Stănciulescu, Sandro Schulze, and Andrzej Wąsowski. 2015. Forked and Integrated Variants in an Open-Source Firmware Project. In *31st International Conference on Software Maintenance and Evolution (ICSME'15)*.

[10] Greg R Vetter. 2007. Open Source Licensing and Scattering Opportunism in Software Standards. *BCL Rev.* 48 (2007), 225.

[11] Shurui Zhou, Ştefan Stănciulescu, Olaf Leßenich, Yingfei Xiong, Andrzej Wąsowski, and Christian Kästner. 2018. Identifying Features in Forks. In *Proceedings of the 40th International Conference on Software Engineering (ICSE)*. ACM Press, New York, NY.