GitHub duplicates identification and filtering

Pol Benats, Maxime Gobert, Loup Meurice, Csaba Nagy and Anthony Cleve

According to [2] the high percentage, 70%, of code duplicates on GitHub is significant enough to consider it in empirical studies. However, the cited report investigates code duplicates at the *file level*, not at the project level, and in the 70% duplicates, they do not differentiate intra- and inter-project clones. For example, the report says that among the 70% code file duplicates, the "empty file" is the most often "cloned" file.

We study individual projects to analyze the use of (multiple) database technologies. File and intra-project duplicates are acceptable in our case when the projects are otherwise different. The same report found that 1–4% of the projects had ten or more exact file duplicates (see Table 5 in the report) when they considered only inter-project clones. Such clones would still not generate noise for us if, otherwise, the projects have differences and should be considered as separate projects.

Potential noise in our dataset can originate from exact *project* duplicates or project clones with minor differences (*e.g.,* a project forked only for the pull request of a bug fix). This is a valid threat. A recent study by Jiang et al. [1] found that only a few forks start the development of new projects, and the root causes of forks are submitting pull requests, fixing bugs, adding features, or only keeping copies of projects. It is possible that our dataset contained such fork projects. However, we notice that forks are unlikely to be starred and, consequently, our quality filter (min. two contributors and min. two stars) was supposed to remove them. If a fork is starred multiple times, it should likely be seen as an individual project. Still, problematic project forks could remain in our filtered dataset, and we do agree that they can impact the validity of our results.

To alleviate the potential noise of project duplicates, we applied an additional filter to identify and remove problematic fork projects.

The dataset of *Libraries.io* has a "fork" flag that points to the source repository of a fork project. We relied on this flag in our filter. We found 3,681 fork projects among the 42,176 projects of our dataset (8.7%). In the initial set of 1.3M projects, there were 136,482 forks (9.8%). Thus, a significant number of fork projects indeed passed our quality filter.

We also notice that, in rare cases, a project can be a duplicate without being marked as a fork. For example, if the source project gets deleted on GitHub, the "fork" flag is set to false for its forks. It can also happen that somebody simply commits a copy of a project. Such cases are extremely rare, and their identification would require a significant effort. Therefore, we consider potential non-forked duplicated projects as a negligible threat and focus on the filtering of fork projects.

We analyzed the 3,681 fork projects to assess whether they are duplicates or individual projects. Consider the following notation: FP is the set of filtered projects (*i.e.,* 42,176 projects), $dbdep(p)$ is the database dependencies of project p, and $p_f \lhd p_s$ means that p_f is a fork project of p_s. We define the projects p_i and p_j ($p_i \neq p_j$) as duplicates, $p_i \simeq p_j$, if one of the following two conditions applies:

1. $p_i \lhd p_j$ or $p_j \lhd p_i$, and $dbdep(p_i) = dbdep(p_j)$; or
2. $\exists p_k\,(\neq p_i, p_j)$, that $p_i \lhd p_k$, $p_j \lhd p_k$ (p_i and p_j are siblings), and $dbdep(p_i) = dbdep(p_j)$.

Next, we identify the maximum DS duplicate sets of projects in FP, where $\forall p_i, p_j \in DS$, $p_i \simeq p_j$, and $p_i, p_j \in FP$. We keep only one project for each DS as follows. There are two types of DS sets. (1) When the source repository is in FP, we simply keep the source project and filter the forks. (2) When the source repository is not in FP, there are only siblings in the filtered set, and we need to decide which sibling we keep even though they are duplicates and have the same database dependencies. In this case, we keep the project with the longest history, *i.e.,* the project that exists in

more dataset versions of libraries.io. If there are more matches, we randomly pick one. The rationale of this decision is to have a better overview of the evolution of the project, which is important for our research question.

Figure 1 presents an overview of the duplicated projects identification process. The leaves in green contain the number of projects that we retain for our survey, others are considered as duplicates and are then removed from our dataset.

Here are two examples of how we remove/keep the duplicated projects:

(1) Source and fork projects are in *FP*: *johnnyworker1012/19wu*[1] is a fork of *19wu/19wu*.[2] Both projects have the following Ruby database dependencies: *activerecord-jdbcpostgresql-adapter*, *mysql2*, *pg* and *sqlite3*. As of July 13, 2021, *johnnyworker1012/19wu* is "13 commits ahead, 873 commits behind *19wu:master*." They are in *FP*, so we keep only the source repository and remove *johnnyworker1012/19wu* from the dataset.

(2) The source is not in *FP*, and multiple siblings are in *FP*: *Gyoung/dubbox*,[3] *Xiaobaxi/dubbox*,[4] *foruforo/dubbox*,[5] and *yihaijun/dubbox*[6] are forked from *dangdangdotcom/dubbox*.[7] Their source is not in the FP set, and the four siblings have all the same Java database dependencies: *redis.clients:jedis* and *com.googlecode.xmemcached:xmemcached*. We keep only one of the four projects. As they are all present in the five versions of the libraries.io dataset, we randomly choose one of them and remove the others.

Overall, 40,609 projects remained in our dataset (out of the 42,176 projects). We fully replicated our study based on the smaller list of filtered projects. We observed that our results remain the same, leading to equivalent trends and percentages compared to the results with duplicates.

[1] https://github.com/johnnyworker1012/19wu
[2] https://github.com/19wu/19wu
[3] https://github.com/Gyoung/dubbox
[4] https://github.com/Xiaobaxi/dubbox
[5] https://github.com/foruforo/dubbox
[6] https://github.com/yihaijun/dubbox
[7] https://github.com/dangdangdotcom/dubbox

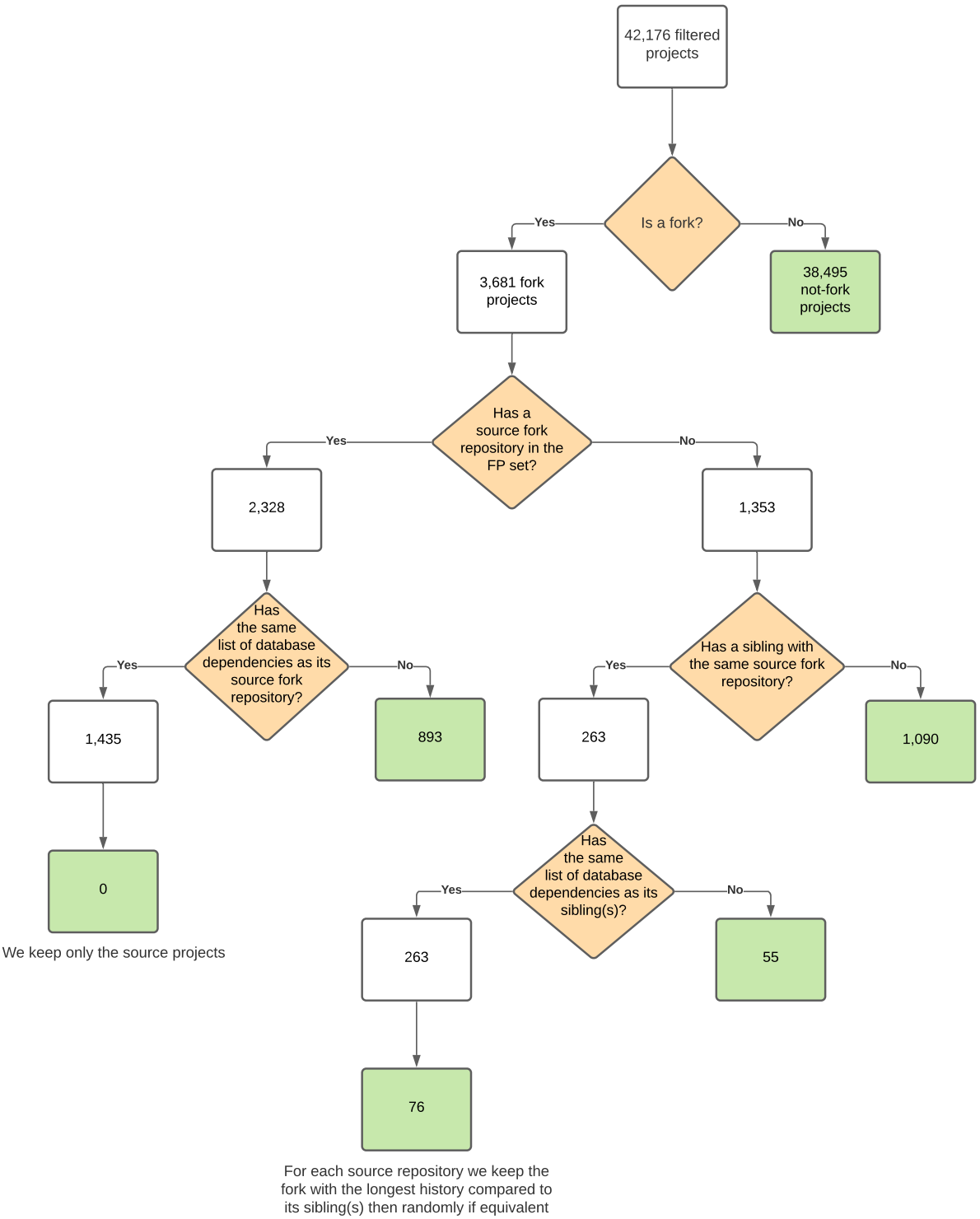

Figure 1: Selection of duplicated projects

Bibliography

[1] Jiang, J., Lo, D., He, J., Xia, X., Kochhar, P.S., Zhang, L.: Why and how developers fork what from whom in GitHub. Empirical Softw. Engg. **22**(1), 547–578 (Feb 2017)

[2] Lopes, C.V., Maj, P., Martins, P., Saini, V., Yang, D., Zitny, J., Sajnani, H., Vitek, J.: Déjàvu: A map of code duplicates on github. Proc. ACM Program. Lang. **1**(OOPSLA) (Oct 2017)