

A Study of Potential Code Borrowing and License Violations in Java Projects on GitHub

Yaroslav Golubev , Maria Eliseeva,
Nikita Povarov, Timofey Bryksin

発表者: 西山

背景

- 現代のソフトウェアの大部分がクローンで構成されていることがわかっている
- 異なるプロジェクト間でコードが再利用されることによってライセンス違反の問題がますます顕著に
- 既存のオープンソースライセンスの数が多いため、開発者がライセンスの条件や違いを十分に理解していないことも原因の一つ

OSSのライセンス

ライセンスのタイプ	主なOSSライセンス	特徴
コピーレフト型	• GNU General Public License (GPL) • Affero General Public License Version 3 (AGPL) • Sleepycat License • The TMate Open Source License	• ライセンステキストの添付が必要 • 改変した（コピー＆ペーストも含む）ソースコードの開示 • 組み合わせて利用した場合、対応する部分のソースコードの開示
準コピーレフト型	• GNU Lesser General Public License (LGPL) • Mozilla Public License (MPL)	• ライセンステキストの添付が必要 • 改変した（コピー＆ペーストも含む）ソースコードの開示
非コピーレフト型	• BSD 2-clause License • BSD 3-clause License • BSD 4-clause License • Apache License 1.1 • Apache License 2.0 • MIT License	• ライセンステキストの添付が必要 • ソースコードを変更したとしても、ソースコードを開示する必要はない

<https://thinkit.co.jp/story/2014/02/03/4804?page=0%2C2>

目的

- **GitHub上のJavaプロジェクトにおけるライセンスの分布を調査し, 小規模なコード借用やライセンス違反の可能性を推定すること**

アプローチ

- **GitHub**から人気のある**Java**プロジェクトの広範なコーパスを収集
- コードクローンを検索してコード断片のレベルでコードの借用やライセンス違反の可能性について分析を行う

アプローチ(データセット)

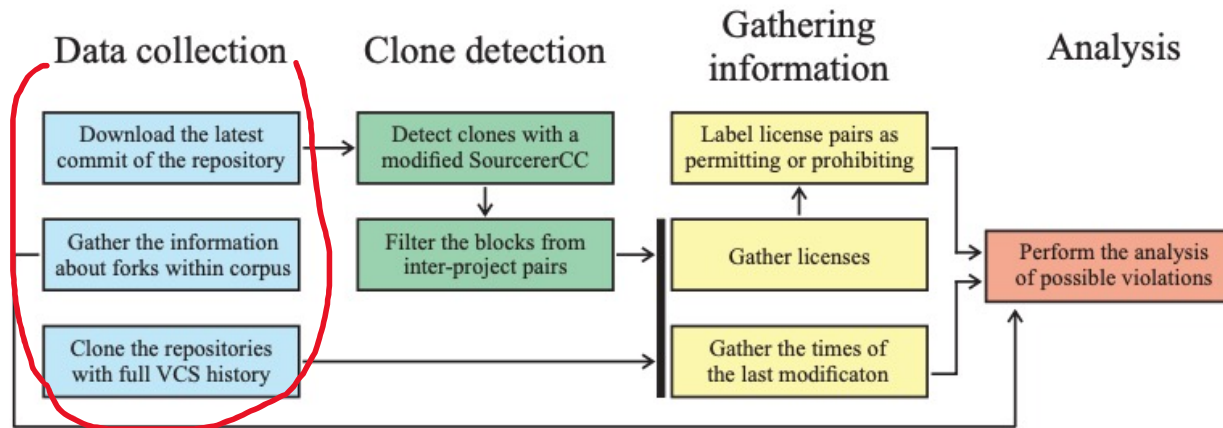


Figure 1: The pipeline of the study.

- Public Git Archive(PGA)で, Javaを1行以上含み, スター数が50以上のプロジェクトをダウンロード
- 借用や違反の可能性を分析するために, VCSの深さを最大に
- 2019年6月1日までの履歴を持つ23,378のJavaリポジトリが集まった
- PGAに備わっている, Go License Detectorを使用してプロジェクトのライセンスの情報も取得

アプローチ(クローンの検出)

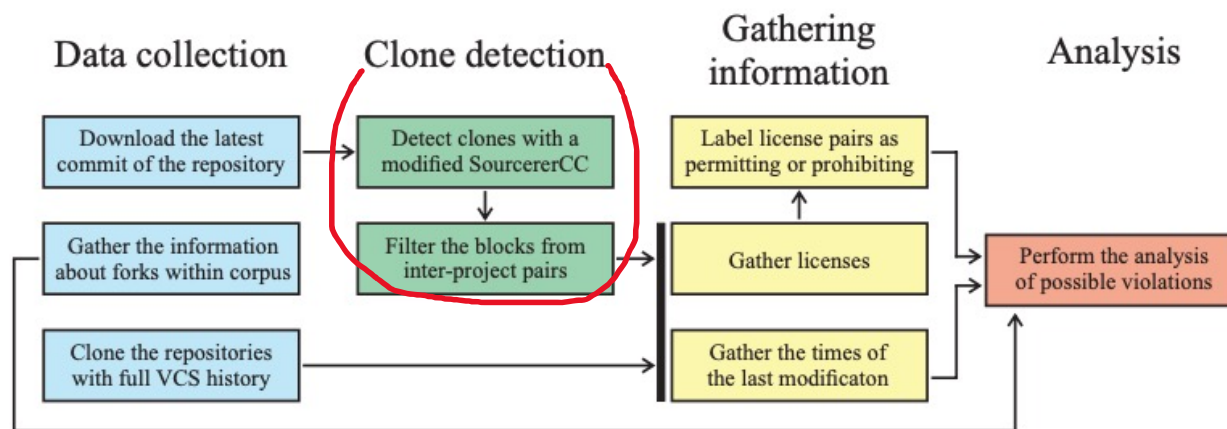


Figure 1: The pipeline of the study.

- SourcererCC[24]を用いてクローンを検出
- クローン検出の出力データはすべてのクローンペアのリスト, ブロックに関する情報で構成

アプローチ(ライセンス情報)

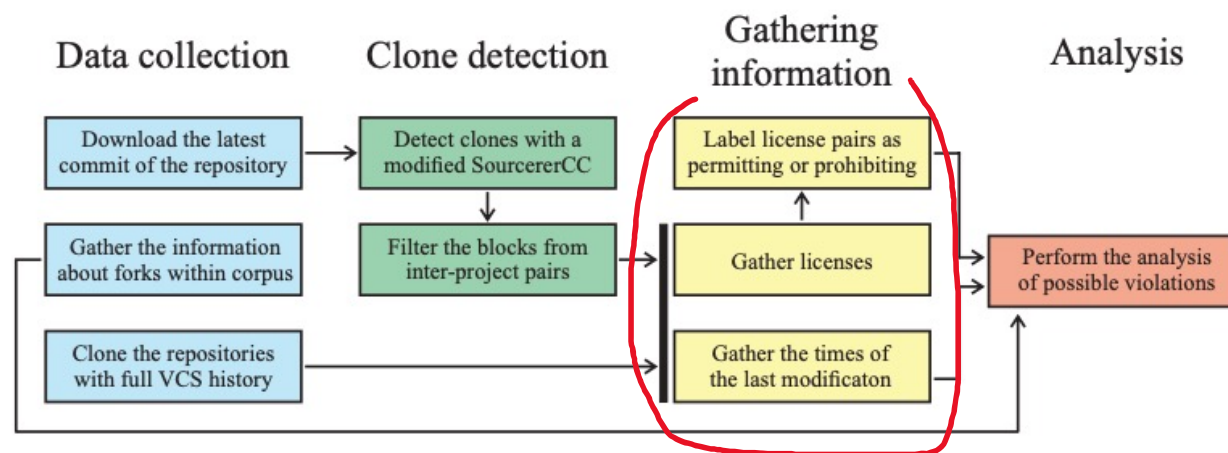


Figure 1: The pipeline of the study.

- ファイル内のライセンスを決定するために, Ninka [9] を使用
- Ninka が PGA とは異なるライセンスを出力することがあるため, PGA と Ninka のリストを作成し手作業で一つ一つ確認
- ブロック・レベルで違反の可能性を検証するため, git blame コマンドで 最終修正日時を収集

アプローチ(分析)

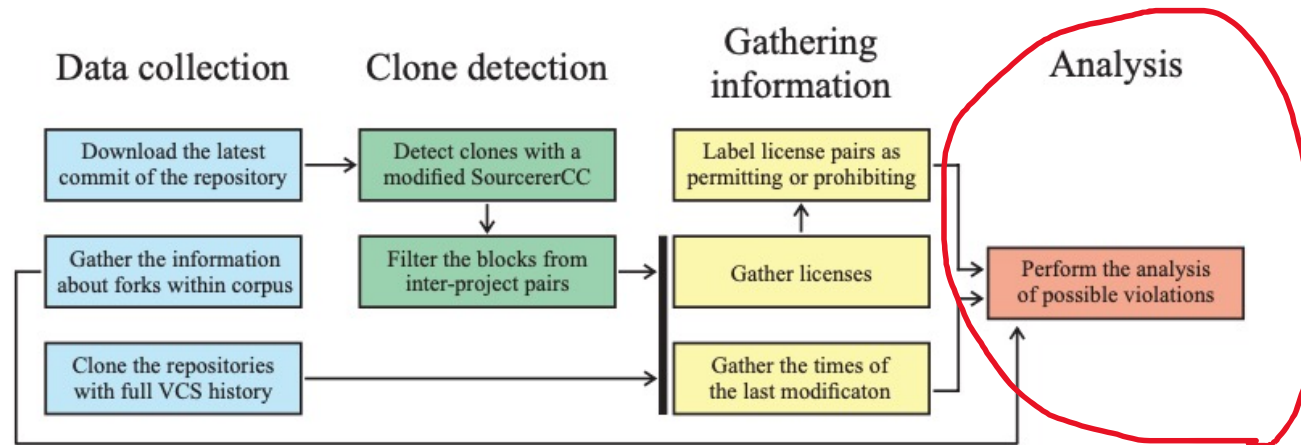


Figure 1: The pipeline of the study.

- クローン検出の結果を, ブロックを頂点, クローンのペアを辺とするグラフ構造として捉えて分析する

違反係数

$$P_{viol} = \frac{N_{prohibited}}{N_{all}}$$

$N_{prohibited}$. . . ブロックの禁止された先行者の数

N_{all} . . . 先行者の総数

クローンのグラフ構造

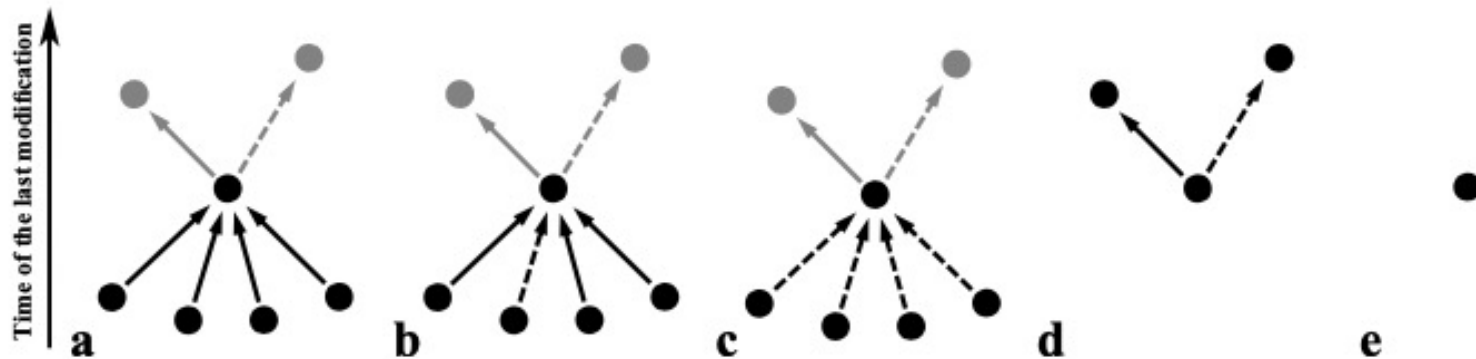


Figure 2: Five types of blocks during the analysis of the neighbors: a) Strong violation; b) Weak violation; c) Legal borrowing; d) Origin block; e) Unique block. The vertical axis indicates the time of the last modification of the blocks, straight lines represent prohibited copying, dashed lines – permitted copying, grey lines indicate that connections are optional for this type.

- a . . . すべての接続がライセンスによって禁止されている場合 → 違反係数が1
- b . . . 接続の一部のみが禁止されている場合
- c . . . 複数の先行者がいるが, すべての接続が許可されている場合
- d . . . 先行はないが後継があるブロック
- e . . . クローンに関与しないブロック

ライセンス違反

- ライセンスのないfiles(GitHub)へのコピー
- コピーレフト・ライセンス(GPLなど)からパーミッシブ・ライセンスへのコピー
- 古いバージョンのライセンス間の特定の非互換性

結果：ライセンスの分布

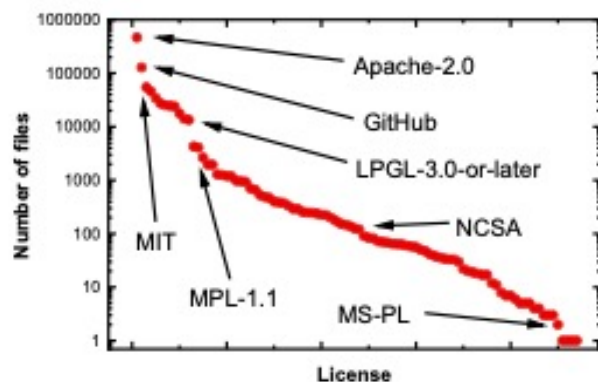


Figure 3: The number of files with various licenses.

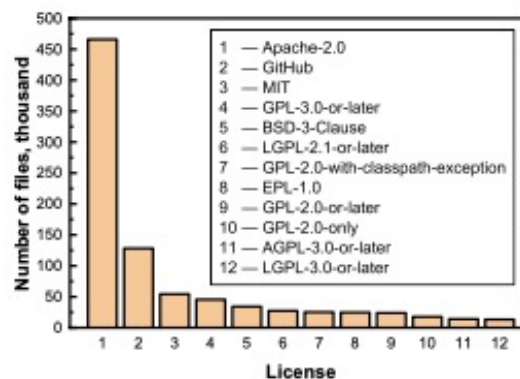


Figure 4: Twelve most popular licenses by the amount of files.

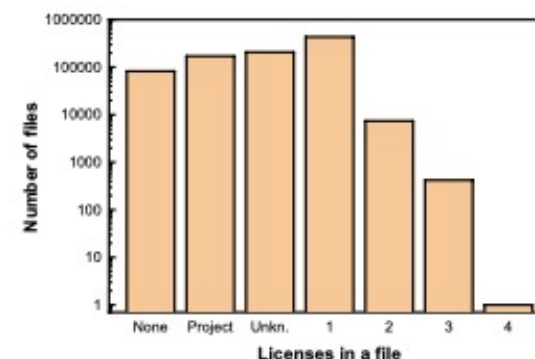


Figure 5: Files by the amount of licenses that cover them.

- Apache-2.0がファイルの半分以上（51.9%）
- 2番目はライセンスがないもの. GitHub上のJavaコードの多くはライセンスのないファイルから来ている

結果:ライセンス違反

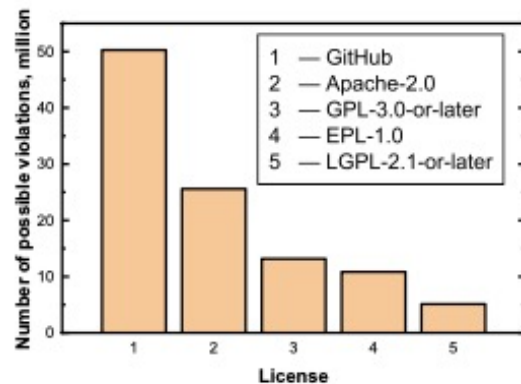


Figure 8: Top five licenses with the most amount of violated blocks of code, i.e. licenses, which act as the source of the largest amount of possible violations.

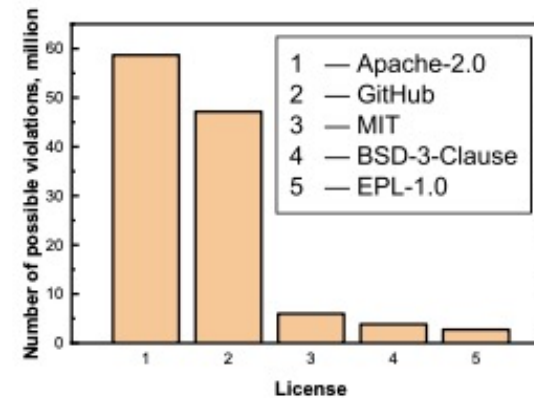


Figure 9: Top five licenses with the most amount of violating pieces of code, i.e. licenses, which act as the receivers of the largest amount of possible violations.

全体で72.8%が合法的な借用であり, 27.2%がライセンス違反の可能性を示していた

- 最も違反されているライセンス
→ Apache-2.0とGitHub
- 最も違反しているライセンス
→ Apache-2.0とGitHubが全体の84%

結果:コード借用

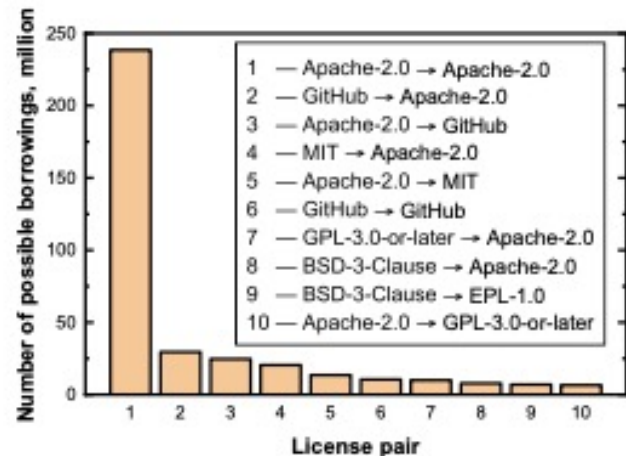


Figure 6: Ten most popular license pairs by possible borrowings.

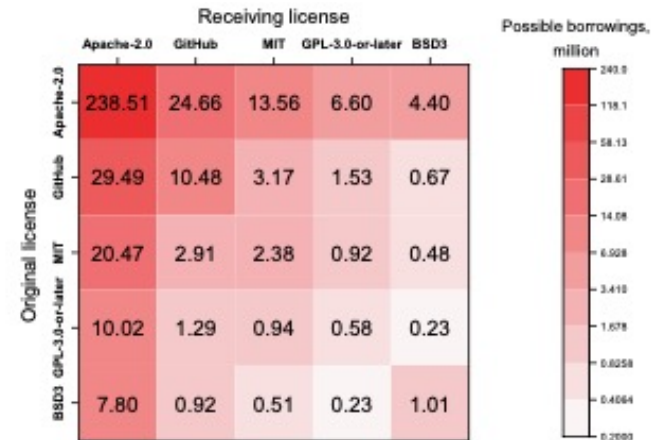


Figure 7: Heatmap of possible borrowings between most frequent licenses with logarithmic coloring, in millions.

- 最も頻度の高い10組のライセンスは, 借用可能な全ライセンスのほぼ80%を占めていた
- 図7はそのうち5組のライセンスについてのヒートマップ
対称性があるが, Apache-2.0では,借用される可能性が若干上回っていた

まとめ

- **GitHub**のJavaコーパスにおいて, ブロックレベルのコード借用やライセンス違反の可能性について調査を行った
- **Apache-2.0, MIT, BSD-3-Clause**などの使用頻度が高かった
- チェックしたファイルの約**9.1%**はライセンスが全く適用されておらず, 違反される可能性が高い

所感

- ファイル単位でライセンスが適用されるのは, 知らなかった
- クローンの検出に2ヶ月強, 要したらしく計算量の多さを感じた
- どれくらい違反しているかの指標は, 自分の研究でも考える必要があると思った