卒 業 研 究 報 告 書

題　目　　StackOverflowのコードスニペットにおける
クローン出現の調査

指導教員　　水野　修　教授

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　09122023

氏　　名　　古塩　卓也

平成31年2月13日提出

StackOverflow のコードスニペットにおける クローン出現の調査

平成 31 年 2 月 13 日　　09122023　古塩　卓也

概　要

StackOverflow(以下 SO) は，ソフトウェア開発者にとって非常に有益な質疑応答を行える Web サイトであり，ソフトウェア開発に関わる多様な質問が，コードスニペットやコメントと共に共有されている．その蓄積されたデータに対して，マイニングや分析を行うことで，ソフトウェア開発に関わる課題を解決する研究が広く行われている．一方で，未解決の課題も多く挙げられている．その一つが，投稿されたコードスニペットにおけるコードクローンの存在の有無である．そこで，本研究ではコードスニペットに対してコードクローン分析を行う手法を確立し，大規模な SO のデータベース SOTorrent に対して適用することで，コードクローンの有無を調査した．具体的には、SOTorrent データセットから Java のコードスニペット群を抽出し，コードクローン検出ソフト CCfinderX によってその数を調査した．

目　次

1. 緒言

SO は，ソフトウェア開発者にとって非常に有益な質疑応答を行える Web サイトであり，ソフトウェア開発に関わる多様な質問が，コードスニペットやコメントと共に共有されている．同時にそのスニペットやコメントは，SO 内は勿論 GitHub を始めとする外部 Web サイトや，個人的なローカルのソフトウェア内にも，全く同じもしくは多少の変更を加えて使われる事が多数ある．このような複製，すなわちクローンの存在はソフトウェア開発に置いて，特に保守を困難にするものと言われている [1]．また，SO 内に置いても，クローンの存在は質問や回答が少なからず重複している事を意味し，その量が余りにも多い場合質疑を行う場として健全であるとは言えない．

そこで，本研究ではコードスニペットに対してコードクローン分析を行う手法を確立し，大規模な SO のデータベース SOTorrent に対して適用することで，その存在を調査した．

本論文の構成を以下に示す．2 章では本研究の目的を，3 章では必要な前知識を，4 章では研究の具体的な手順と方法を述べる．5 章に調査の結果を示し，6 章で考察，7 章で結言とする．

2. 研究の目的

本研究の最終目的は，SO 内に存在する全てのコードスニペットの取得，及びその中のクローン数を調査し，コミュニティに対してクローンの情報を提供することである．しかし，全てのスニペットを取得分析することは，データ量及びかかる時間のどちらの点からも非常に困難である．そこで，本研究ではクローンを SO 内から取得分析するための手順を確立させることを目指した．また，提案する手順が有効であることを示すために，次のような研究設問に対する分析を行った．

RQ1: Java のコードスニペット・クローンは，年間でいくつ存在するか

RQ2: RQ1 の区間を月単位で分けた場合，そのスニペットとクローンの割合はどう変わるか

3. 用語

本章では，論文で用いる用語について示す.

3.1 コードスニペット

コードスニペット (以下スニペット) とは，プログラムコードの一部分・断片を指し，多くの場合，多少の変更を加えることで再利用可能な物が多い．モジュールや関数と呼ばれるものは再利用可能なスニペット例と言える.

3.2 コードクローン

コードクローン (以下クローン) とは，プログラムコードの中で論理的な構造が非常に似ている，もしくは完全に同じブロックの事を言う．上記のスニペットの性質から，スニペットを用いたコードはその部分がクローンとなる可能性が高い.

3.3 StackOverflow

StackOverflow (注 1) とは，情報技術，特にプログラミング技術に関するコミュニティサイトである．プログラミング初心者からソフトウェア開発者・研究者まで幅広く質疑応答が行われており，そのドキュメントやスニペットは誰でも閲覧することが出来る．StackOverflow 自体を対象にした研究も盛んに行われており，SOTorrent はその一つである.

3.3.1 SOTorrent

SOTorrent とは，StackOverflow の公式データダンプ [2] を基にして作成された，オープンなデータセットである [3]． StackOverflow 自体の研究をするにあたって，公式のデータダンプは，一つの投稿単位での管理がなされており，ドキュメント・スニペット単位でのバージョン変化を追うことが困難であるという背景から作られている．2018 年 8 月 28 日にリリースされた SOTorrent バージョンで，SO 公式の 2008 年 7 月

(注 1): https://stackoverflow.com/

31 日の最初の投稿の作成から 2018 年 6 月 3 日の最後の編集までに至る，40,606,950 の質問と回答すべてのバージョン履歴が含まれている [3].
SOTorrent のスキーマを図 3.1 に示す．図 3.1 においてグレーで表示されているテーブルは SO 公式のものであり，ブルーで表示されているテーブルが追加されたテーブルである [2][4].

3.4 SQL

SQL とは，データに対して問い合わせを行う為に用いられるクエリ言語の一つである．本研究では SOTorrent のデータを展開する為，MySQL (注 2)をデータベースとして使用し，SQL を使用してデータの挿入や抽出を行う.

3.5 Guesslang

Guesslang とは，与えられたソースコードが何の言語で書かれているかを推測・判定する Python ライブラリである．機械学習アプリケーションであり，分類機として線形分類機と 3 層ニューラルネットワークを使用している [5]．対象言語は 20 個あり，本研究で対象とする Java も含まれている．
オープンソースであり，実装が GitHub に公開されている．(注 3)

3.6 CCfinderX

CCfinderX とは，神谷ら [6] によって開発されたコードクローン検出ツールである．(注 4) 以下に示すような様々なオプションがある [1].

- メトリック LEN: 検出されるクローンの長さの最小値．長さはトークン単位で測られる．デフォルトは 50.
- メトリック TKS: 検出されるトークンの種類数の最小値．デフォルトは 12.
- 検出範囲: ファイル間クローンを検出するか，ファイル内クローンを検出するかどうか．デフォルトはどちらも検出する．

(注 2): https://www.mysql.com
(注 3): https://github.com/yoeo/guesslang
(注 4): http://www.ccfinder.net/ccfinderxos-j.html

なお，公式の最新バージョンである CCfinderX は 2010 年で開発が止まっており，当時の実行環境 (Win32, Ubuntu9.1) を用意することが困難である理由から，GitHub 上で第 3 者によってフォークされているものを使用した．(注 5)

(注 5): https://github.com/radekg1000/ccfinderx

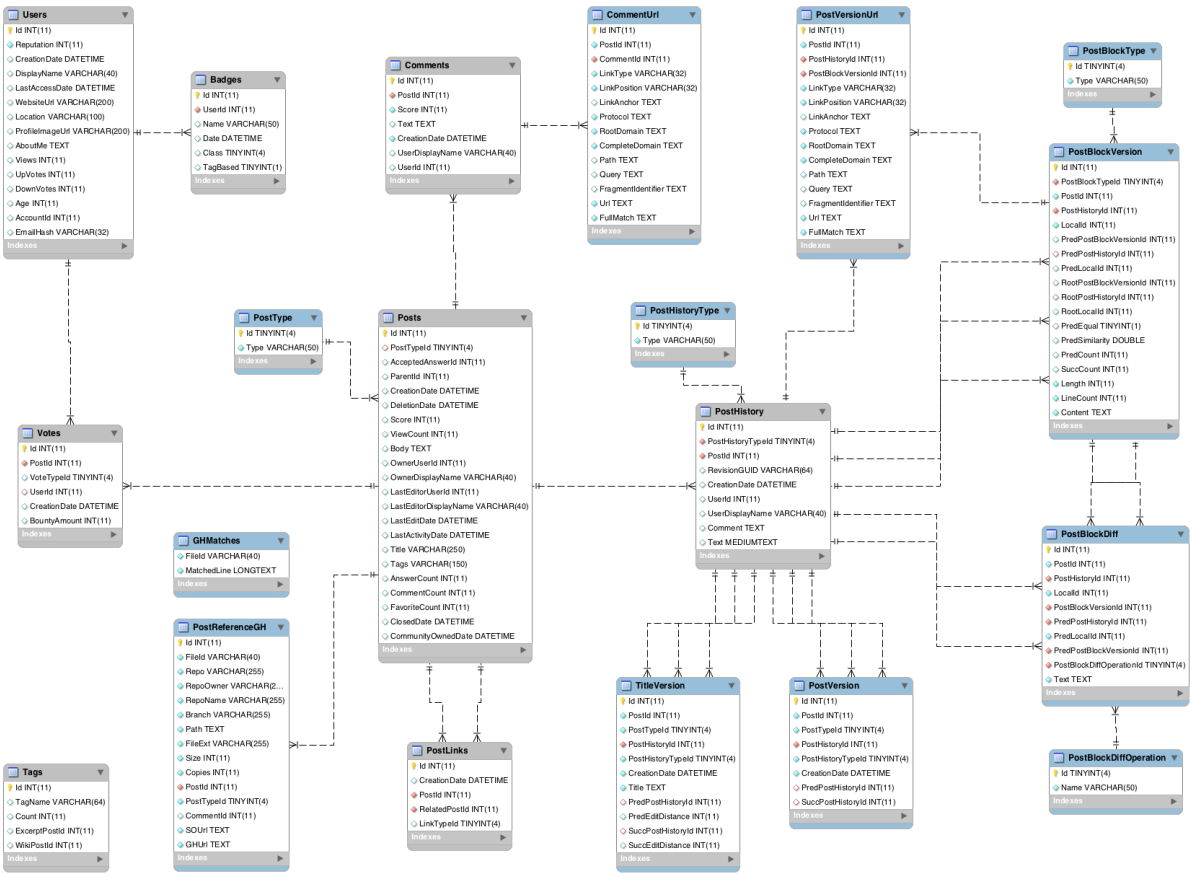

図 3.1　SOTorrent database schema

4. 実験

本章では，実験を行った環境や手順を示す．

4.1 データベース構築

SOTorrent ダンプを Zenodo (注 6) よりダウンロードしデータベースをローカルに構築する．構築には推奨されている MySQL（Ver5.7）を用いた．構築手順もまた示されており，詳細は付録 A にある．

実験で使用した SOTorrent は，「Version 2018-09-23」であり，3 章で述べたものよりデータの最終期間が 2018 年 9 月 2 日まで増えている．なお，SOTorrent は Google Big Query (注 7) 上にもオンラインで展開されており，そちらを使用する場合データベースの構築は不要となる．

4.2 コードスニペットの取得

構築したデータベースに対し SQL を用いて，目的のスニペット群を取得する．取得形式は TSV ファイルであり，次の前処理部に引き渡す．取得するスニペットの条件とその目的を以下に示す．

対象: 質問及びそれに対する回答全てに含まれるコードスニペット

条件: 投稿データに Java タグが含まれており，その投稿差分の中で最新のもの

Java を選んだ理由は，投稿数が Javascript に次いで 2 番目に多く，取得条件として適切であると判断出来ること．筆者が Javascript よりも Java に知識があることによる．

期間: 2017/9 2018/8 の 1 年間及び，その期間の 1 か月ごと

1 年間の区間の定め方は，最新データから 1 年間遡るとして決定した．

今回与えた取得クエリの 1 例を List4.1 に示す．内容は「2018 年 8 月中に Java タグを含む投稿からスニペット部分のみ抽出し，その最新バージョンとバージョン ID を取得する」というものである．

(注 6): https://zenodo.org/record/1434285
(注 7): https://bigquery.cloud.google.com

Listing 4.1 スニペット取得クエリ

```
SELECT pb.Id, Content FROM PostBlockVersion AS pb
  INNER JOIN
  (SELECT MAX(pb.Id) AS Id FROM PostBlockVersion AS pb
    INNER JOIN Posts AS p ON pb.PostId = p.Id
        AND Tags LIKE '%<java>%'
   WHERE CreationDate BETWEEN '2018-08-01 00:00:00' AND
    '2018-08-31 23:59:59'
    AND PostBlockTypeId = 2 /* Code Block */
  GROUP BY RootPostBlockVersionId)
AS Lid ON pb.Id = Lid.Id;
```

4.3 前処理

取得したデータを CCfinderX での解析にかける為に，Java のスニペットのみを抽出し，1 つずつの java ファイルとして保存する．処理の流れを以下に示す．

1. スニペットを含んだ TSV ファイルから各スニペット部分を全て抽出し，1 つずつの java ファイルとして保存する．
2. 得られた java ファイルを Guesslang により言語判定を行い，Java コードと判定されたものだけに絞り込む．

4.4 解析

CCfinderX を用いてクローンの判定を行う．各設定はデフォルトのものを使用している．なお，前処理を行っていても CCfinderX が処理できない (解析用プリプロセス不能) ファイルが一定数出現するため，一度ファイルを CCfinderX に通した後，処理不能であったものを省いている．解析の流れを以下に示す．

1. 対象の java ファイル群を CCfinderX に通す．
2. 処理できないファイルがあれば省く．無ければ結果を得て終了．
3. 1 に戻る．

ここで対象の java ファイル群とは前処理までで得られた,「年間の java ファイル群全て」と「月ごとの java ファイル群」を指す．

5. 結果

本章では，実験によって得られた結果を示す．

実験を通して得られた具体的なクローンの例を図 5.1 及び 5.2 に示す．図 5.1 は 1 つのスニペット内に存在するコードクローンの実例であり，水色のハイライト部分がそうである．また，図 5.2 は 2 つのファイルにまたがって検出されたコードクローンの実例であり，灰色のハイライト部分がそうである．

以下で，実験対象スニペット数は，実験の過程を通して最終的に残った検証対象ファイルの数である．クローン率はクローン数を実験対象スニペット数で割ったものである．

5.1 実験結果 1

年間 (2017/09/01/00:00〜2018/08/31/23:59) における，全スニペット数・実験対象スニペット数・クローン数・クローン率をそれぞれ表 5.1 に示す．全スニペット数は 298,958 個，ここから対象として抽出したものが 176,283 であった．また，実験対象スニペットからコードクローンを抽出した結果，33,365 個を発見した．

5.2 実験結果 2

月単位に分けてコードクローンの割合を測ったもの (2017/09/01/00:00 2017/09/30/23:59,...,2018/08/01/00:00 2018/08/31/23:59) を表 5.2 に，そのグラフを図 5.3 に示す．クローン率はおよそ 0.15 0.19 の間を推移している．

なお，表 5.2 のクローン数の合計は年間のクローン数と一致しない．これは月別では月をまたいだクローンが検出できていないことによる．

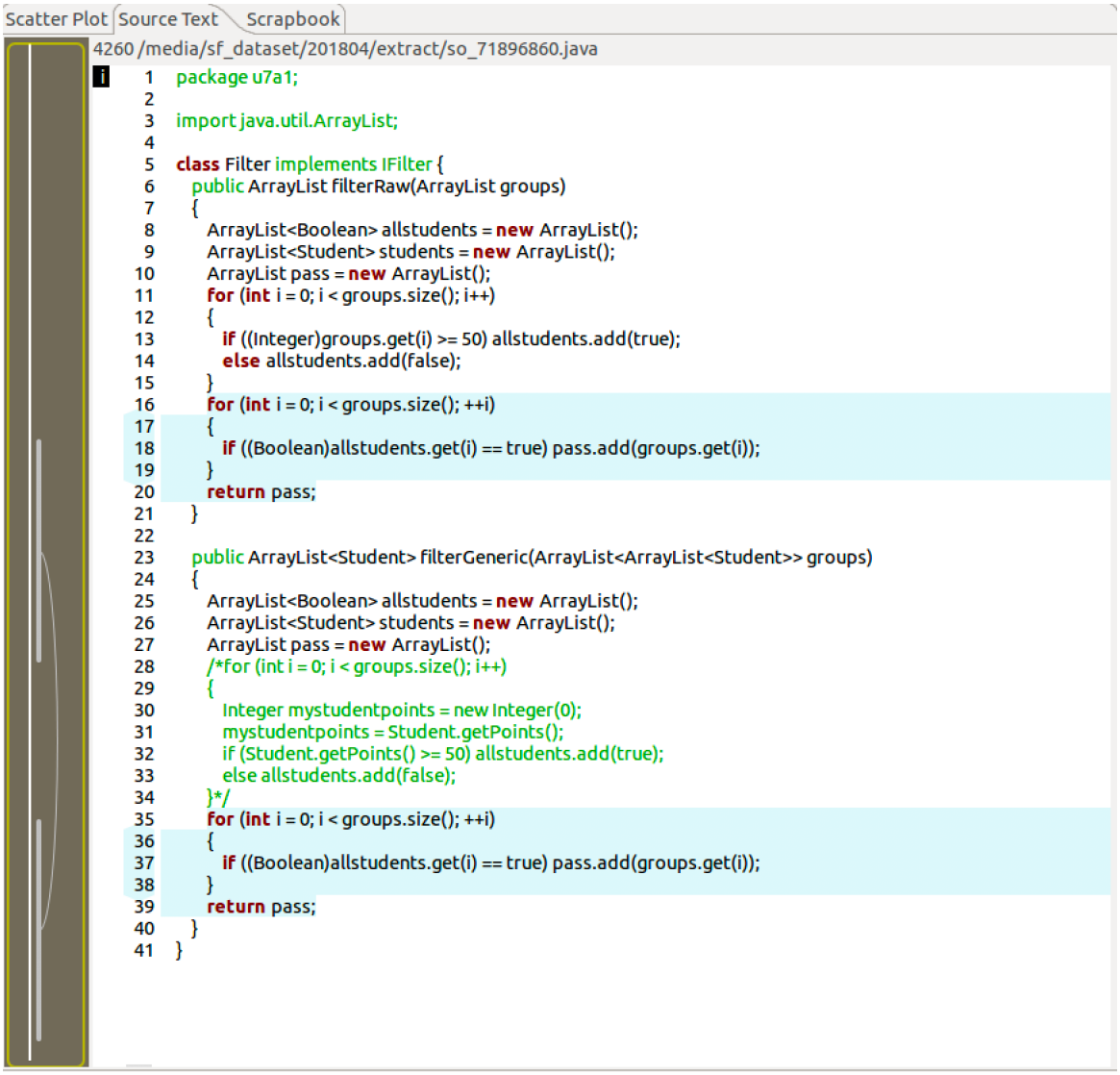

図 **5.1**　ファイル内クローンの例

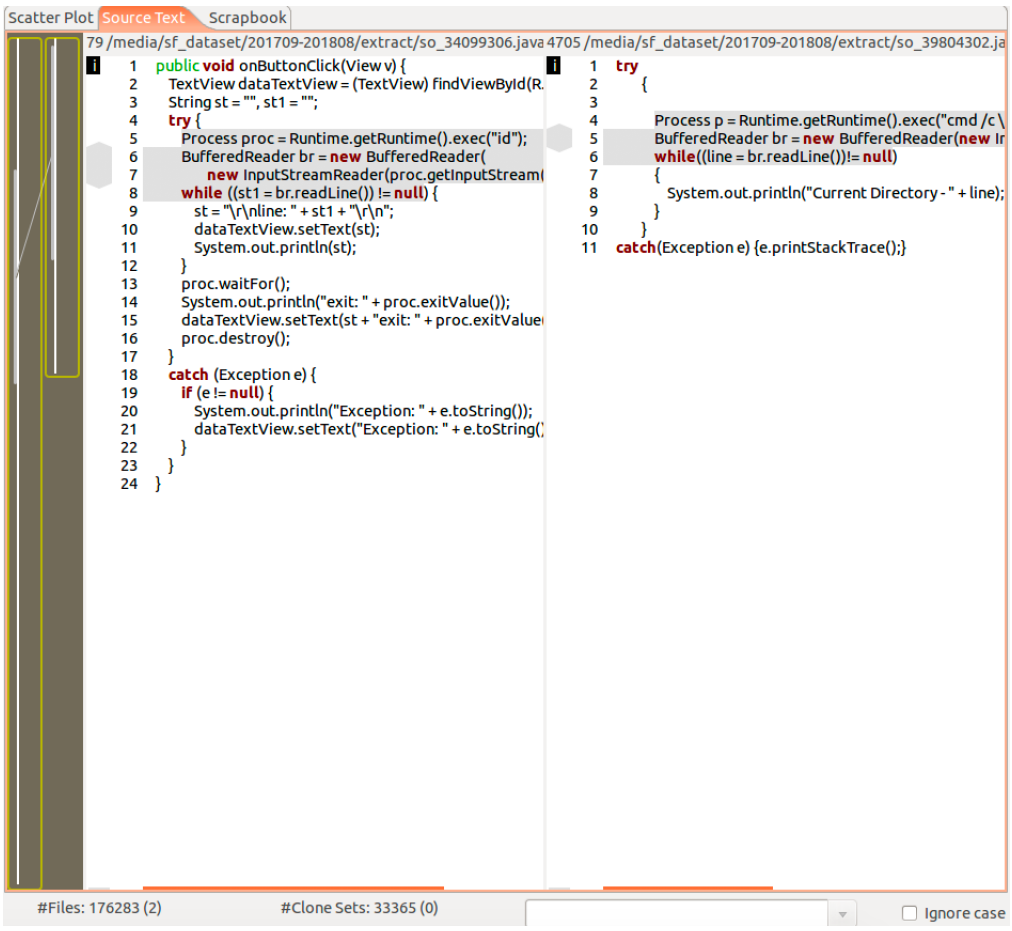

図 **5.2**　ファイル間クローンの例

表 5.1　年間クローン数

期間	全スニペット数	実験対象スニペット数	クローン数	クローン率
201709-201808	298958	176283	33365	0.1893

表 5.2　各月におけるクローン数

期間	全スニペット数	実験対象スニペット数	クローン数	クローン率
201709	24611	14458	2153	0.1496
201710	27223	16153	2760	0.1709
201711	27737	16478	2824	0.1714
201712	23760	14202	2347	0.1653
201801	23788	13836	2472	0.1787
201802	23235	13750	2655	0.1931
201803	25921	15287	2566	0.1679
201804	26369	15858	2721	0.1716
201805	26230	15638	2849	0.1821
201806	23142	13527	2307	0.1705
201807	23010	13466	2452	0.1829
201808	23932	13762	2226	0.1617

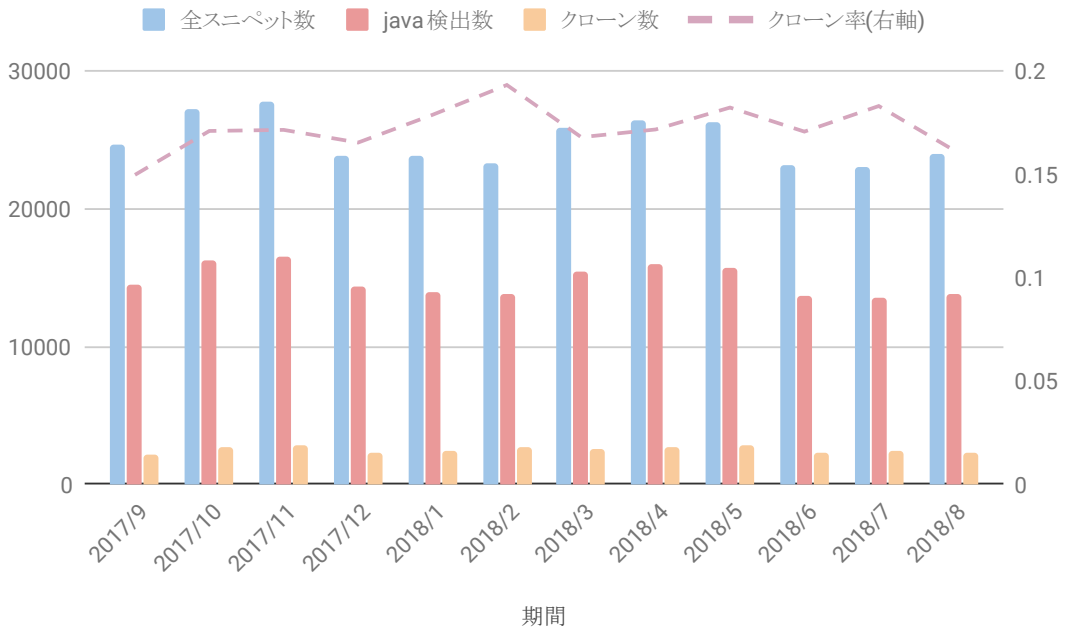

図 5.3　各月におけるスニペット数とクローン検出数

6. 考察

6.1 研究設問について

第 2 章で提示した設問に対して，第 5 章で得られた結果を基に考察していく．

RQ1: Java のコードスニペット・クローンは，年間でいくつ存在するか

この答えに関しては表 5.1 の通りである．Java スニペットの約 19%はクローンであることが分かった．一方で，Java のスニペットを取得する際には，少なくないノイズが存在することがわかった．

RQ2: RQ1 の区間を月単位で分けた場合，そのスニペットに対するクローンの割合はどう変わるか

表 5.2 及び図 5.3 より，クローン率は月毎であまり変化が見られず，年間を通して同程度である．また，月毎のクローン率の平均を取ると 0.1721 であり，年間クローン率 0.1893 と大きく離れてはいない．つまり，RQ2 に関して答えは，クローンの割合は，年単位と月単位ではその期間の大きさによらず，あまり変わらない．

この事から，一つの Java スニペットに着目した時，それと同じ月で見ても含む年内で見ても，一定の確率でそのクローンが存在する．これは，Java というコンテンツについて似通った質問・もしくは解決策が，その期間の大きさによらず，一定の確率で発生しているとも考えられる．更に，StackOverflow 内の Java のタグが付けられた投稿数は，1,510,914 で Javascript の 1,756,962 に次ぐ 2 番目の多さであり，(2019/02/09 現在) 内容が重複しやすいことが推測できる．これについては，同様に投稿数が多い Javascript や C#に関して同じ調査を行うことで，検証できるだろう．

6.2 妥当性への脅威

RQ1 でも述べた通り，今回の結果にはノイズの混入及び取得漏れが存在しており，それを以下に示し考察していく．

1. Guesslang によって Java と判定された Java で無いコード・出力データ等の文字列
2. Guesslang によって Java と判定されなかった Java のコード
3. CCfinderX が処理できない Java のコード

全てに共通して，これらは使用する手法とツールに起因するものである為，それ自体のソースコード・パラメータを調整する，もしくは別の手法を用いることで改善が可能である．

1 及び 2 に関して，実際に結果に影響を及ぼすのは「Java スニペットではないのに Java スニペットと判定され，その中でクローンが検出される場合」と「Java スニペットであるのに Java スニペットではないと判定され，クローンを含んでいたのに解析されなかった場合」である．現在の Guesslang の正確性は 90%以上と非常に高いことから [5]，判定を間違える量よりも間違えた際のデータの重み (影響を及ぼすパターンの出現確率) が重要と言える．この点で 2 に関しては，漏れたデータがクローンである可能性は結果のクローン率からも分かる通り低い．また，クローンであったとしてそれは通常の量である．一方で，1 に関しては，解析された時点で異常な量のクローンを検出してしまう出力ファイルが含まれている．(図 6.1)

3 に関しては，処理不能であった Java スニペットは実験対象スニペット数に対して非常に少なく (表 6.1)，解析不可率は平均で約 0.5%であることから，結果に大きく影響を及ぼすクローン数が存在する可能性は低いと考えられる．

これより，最も影響を及ぼしているのは 1 と言える．

表 **6.1**　各月のスニペットに対する **CCfinderX** の解析不可数とその率

期間	実験対象スニペット数	解析不可数	解析不可率
201709	14387	71	0.0049
201710	16153	103	0.0063
201711	16478	91	0.0055
201712	14202	78	0.0054
201801	13836	76	0.0054
201802	13750	83	0.0060
201803	15287	113	0.0073
201804	15858	77	0.0048
201805	15638	88	0.0056
201806	13527	99	0.0073
201807	13405	61	0.0045
201808	13762	57	0.0041

図 **6.1**　クローンとして検出されるログファイルの例

7. 結言

本研究では，StackOverflow 内に存在する Java スニペットを SOTorrent を使用して抽出し，CCFinderX を通すというクローンの出現調査の手法を確立した．また，その出現数や期間による割合を導いたことで，Java スニペットは期間によらず一定の割合でクローンが存在することを示した．同時に，StackOverflow 内の質問と回答に一定の割合での重複が存在するという新たな仮説が生み出された．

今後の課題として，先の仮説を考察する為別の言語・別の区間設定によるクローン調査を行う事，妥当性を高めるにデータにノイズが入りにくい手法を模索する事が挙げられる．

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系水野修教授に厚く御礼申し上げます．SOTorrent において，貴重なデータを公開されている，Sebastian Baltes 氏に深く感謝致します．本報告書執筆にあたり貴重な助言を多数頂きました，本学情報工学専攻近藤将成先輩，國領正真先輩，塩津拓真先輩，情報工学課程若林奎人君をはじめとする，ソフトウェア工学研究室の皆さん，学生生活を通じて著者の大きな支えとなった，学生サービス課諸角恒宜副課長，藤川洋子教授，藤井幹世カウンセラー，木村智史ピアチューターをはじめとする，アクセシビリティセンターの皆様，公私問わず様々な面で相談に乗ってくれた家族や友人に深く感謝致します．

参考文献

[1] “Ccfinderx ver. 10.2 ドキュメント,” http://www.ccfinder.net/doc/10.2/ja/index.html. (Accessed on 02/11/2019).

[2] “Stack exchange data dump : Stack exchange, inc. : Free download, borrow, and streaming : Internet archive,” https://archive.org/details/stackexchange/. (Accessed on 02/10/2019).

[3] S. Baltes, C. Treude, and S. Diehl, “Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets,” arXiv preprint arXiv:1809.02814, 2018.

[4] S. Baltes, L. Dumani, C. Treude, and S. Diehl, “Sotorrent: reconstructing and analyzing the evolution of stack overflow posts,” Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018, pp.319–330, 2018.

[5] “Guesslang documentation — guesslang 0.9.4 documentation,” https://guesslang.readthedocs.io/en/latest/. (Accessed on 02/11/2019).

[6] T. Kamiya, S. Kusumoto, and K. Inoue, “Ccfinder: a multilinguistic token-based code clone detection system for large scale source code,” IEEE Transactions on Software Engineering, vol.28, no.7, pp.654–670, July 2002.

[7] “Sotorrent dataset — zenodo,” https://zenodo.org/record/2273117. (Accessed on 02/10/2019).

付録 A.　SOTorrent データベースの構築手順

以下は Zenodo の README.md より引用である [7].

1. Unzip all CSV and XML files: 'gunzip *.gz'
2. Execute the SQL script '1_create_database.sql' in your database client (tested on MySQL 5.7) to create the database and tables for the SO dump.
3. Edit the SQL script '2_create_sotorrent_user.sql' to choose a password for the sotorrent user and execute the script to create the user.
4. Execute the SQL script '3_load_so_from_xml.sql' to import the SO dump from the XML files (please use the XML files provided by us, they are processed to be compatible with MySQL).
5. Execute the SQL script '4_create_indices.sql' to create the indices for the SO tables.
6. Execute the SQL script '5_create_sotorrent_tables.sql' to add the SOTorrent tables to the SO database.
7. Execute the SQL script '6_load_sotorrent.sql' to import the SOTorrent tables from the CSV files.
8. Execute the SQL script '7_load_postreferencegh.sql' to import the references from GitHub projects to SO from the CSV file PostReferenceGH.csv.
9. Execute the SQL script '8_create_sotorrent_indices.sql' to create the indices for the SOTorrent tables.