

Studying the Change Histories of Stack Overflow and GitHub Snippets

Saraj Singh Manes, and Olga Baysal,

2021 IEEE/ACM 18th International Conference on Mining Software
Repositories (MSR), 2021, pp. 283-294

発表者: 井町 瑠貴

背景

Stack Overflow (SO) はソフトウェア開発者に人気のあるQ&Aフォーラムで、コピー可能なコードスニペットを大量に提供している

一方、GitHub (GH) は共同開発プラットフォームであり、開発者はSOのコードをGHプロジェクトで再利用することが多い

背景

最近の研究では、コードスニペットが有害である可能性があり^[7]、ライセンス違反^[8]やセキュリティ脆弱性の移行^{[9][10]}につながることを示されている

SOのスニペットがどのように再利用されるかを理解することは、バグや脆弱性の移行を軽減するために重要である

[7] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, “Toxic code snippets on stack overflow,” IEEE Transactions on Software Engineering, pp. 1–1, 2019.

[8] L. An, O. Mlouki, F. Khomh, and G. Antoniol, “Stack Overflow: A code laundering platform?” in International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017, pp. 283–293.

[9] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, “Stack Overflow Considered Harmful? The Impact of Copy Paste on Android Application Security,” in IEEE Symposium on Security and Privacy (SP), 2017, pp. 121–136.

[10] M. Verdi, A. Sami, J. Akhondali, F. Khomh, G. Uddin, and A. Karami Motlagh, “An Empirical Study of C++ Vulnerabilities in Crowd-Sourced Code Examples,” IEEE Transactions on Software Engineering, pp. 1–1, 2020.

目的

SOおよびGH上のスニペットの変更,
および共同変更パターンを調査する
大規模な研究を行うこと

また, 本研究では, 進化をスニペットの変更履歴と
定義している

リサーチクエスト

RQ1: スニペットはどの程度進化しているのか

RQ2: スニペットはどのように進化するのか

RQ3: 誰がスニペットの進化を促進しているのか

RQ4: SOのスニペットと再利用されたコードの
スニペットはどの程度まで同じ変化をするか

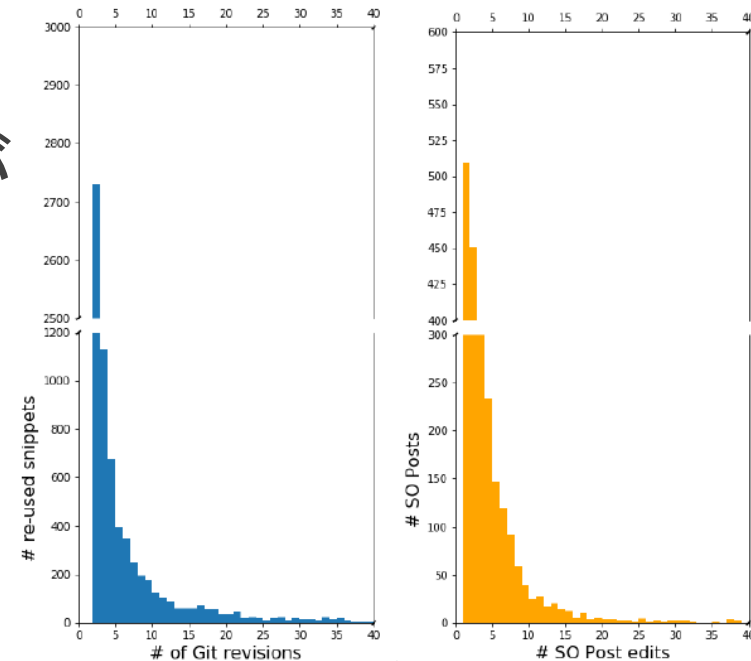
データセット

- SOTorrent^[6]とGHTorrent^[31]を使用し、GHCodeSnippetHistoryデータセットを作成
- 22,900のプロジェクト, 4,634のSO投稿にマッピングされた33,765のSO参照, 合計73,322のコミットが含まれている

[6] S. Baltes, C. Treude, and S. Diehl, "SOTorrent: Studying the origin, evolution, and usage of stack overflow code snippets," in Proc. of the International Conference on Mining Software Repositories, 2019, pp.191–194.
[31] G. Gousios, "The GHTorrent Dataset and Tool Suite," in Proc. of the Working Conference on Mining Software Repositories, 2013, pp. 233–236.

RQ1: スニペットはどの程度進化しているのか

- GHでは, コードスニペットの 22%が進化
- SOでは, 76.3%のスニペットが進化
- 平均して SOのスニペットは, GHで進化するスニペットに比べて変更が少ない



RQ2: スニペットはどのように進化するのか

TABLE V: Edit characteristics for Stack Overflow snippets.

Edt. #	Major Change Type	2nd Major Change Type	Time (in H,D,Y)			Change Size	
			Min	Max	Mdn	Text	Code
0	TEXT&CODE	TEXT	–	–	–	3	8
1	TEXT	TEXT&CODE	< 1H	9.8Y	3.5D	2	2
2	TEXT	CODE	< 1H	10Y	143D	2	1
3	TEXT	TEXT&CODE	< 1H	9.1Y	149D	2	1
4	TEXT	CODE	< 1H	10.3Y	111D	2	1
5	TEXT	TEXT&CODE	< 1H	8.3Y	144D	2	1
6	TEXT	CODE	< 1H	7.3Y	110D	2	1
7+	TEXT	CODE	< 1H	6.6Y	6.2D	1	2

- SOのスニペットは平均して, TEXTに2行, CODEに1行の変更が行われている
- 主要な変更タイプは, TEXTである

RQ2: スニペットはどのように進化するのか

TABLE VI: Revision characteristics for GitHub code snippets.

Rev. #	Major Change Type	2nd Major Change Type	Time (in H,D,Y)			Change Size [LOC]
			Min	Max	Mdn	
0	ADD	MODIFY	–	–	–	930
1	MODIFY	DELETE	< 1H	5Y	11D	30
2	MODIFY	RENAME	< 1H	4.7Y	10D	21
3	MODIFY	DELETE	< 1H	5.8Y	7.8D	23
4	MODIFY	DELETE	< 1H	5.6Y	6.7D	23
5	MODIFY	DELETE	< 1H	4.3Y	6D	23
6	MODIFY	RENAME	< 1H	3.7Y	6.9D	26
7+	MODIFY	DELETE	< 1H	4.9Y	4.3D	22

- GHのコードスニペットの変更サイズは24LOCである
- 主要な変更タイプは, MODIFIEDである

RQ3: 誰がスニペットの進化を促進しているのか

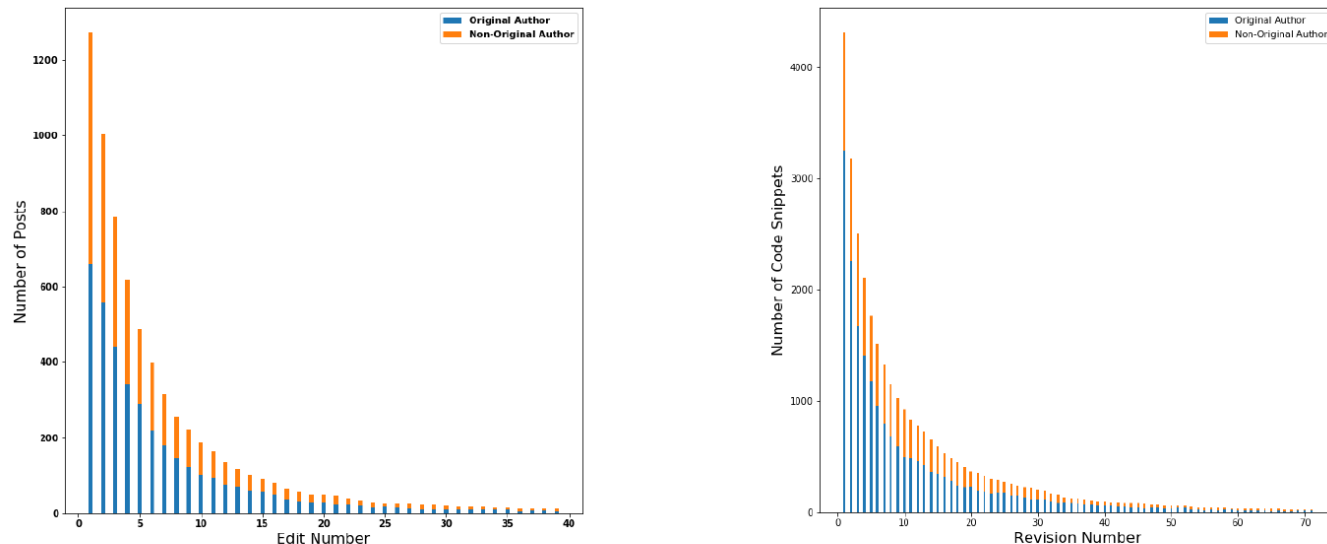


Fig. 6: Contributors on Stack Overflow (left) and GitHub (right).

どちらも、コンテンツの作成者が、コンテンツの維持と進化の主要な貢献者であり続けている

RQ4:
SOのスニペットと再利用されたコードの
スニペットはどの程度まで同じ変化をするか

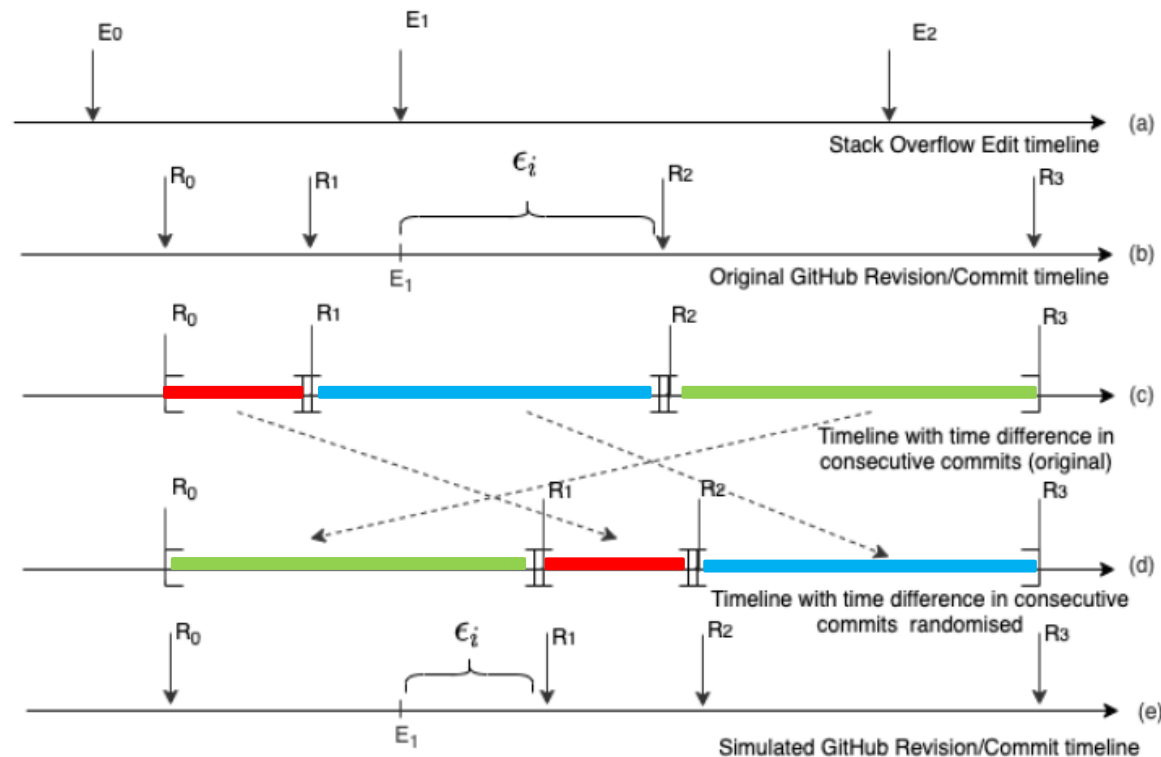


Fig. 3: Generating a simulated time-series of GitHub commit activities.

RQ4:

SOのスニペットと再利用されたコードのスニペットはどの程度まで同じ変化をするか

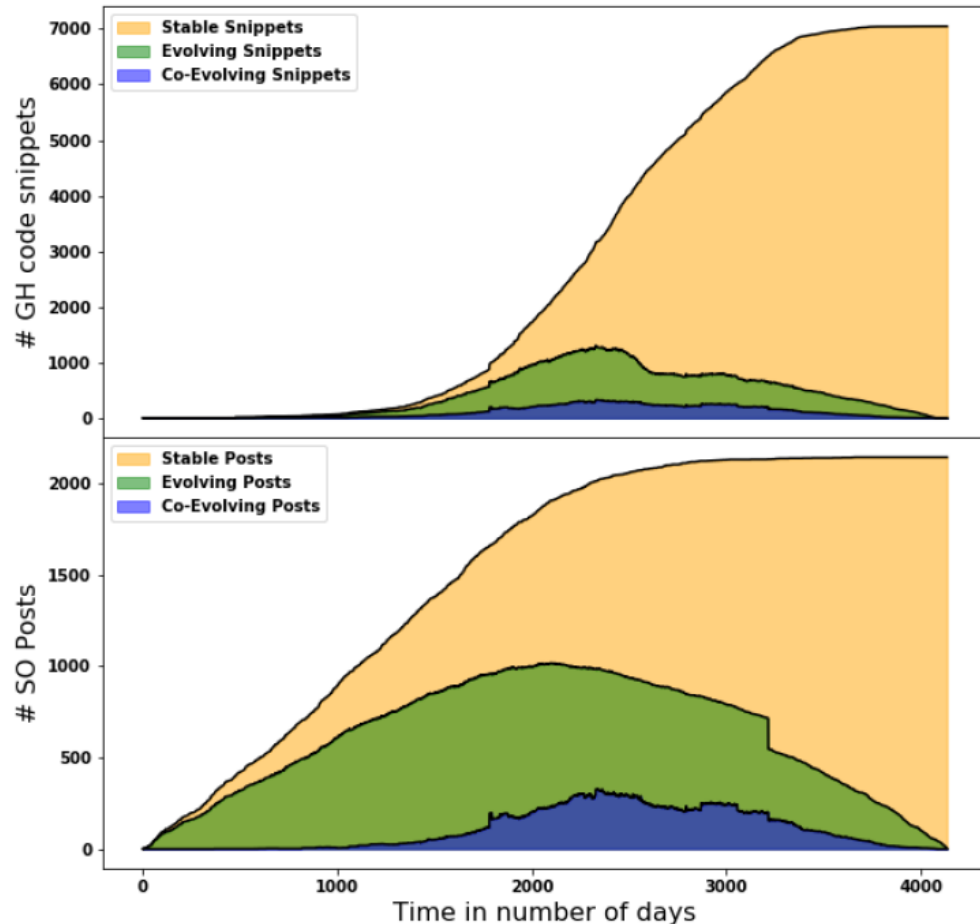
黄: スニペットの総数

緑: 進化する
スニペットの数

青: 共同変化した
スニペットペアの数
(上下で共通)

期間

2008/8/5 ~ 2019/9/1



RQ4:

SOのスニペットと再利用されたコードのスニペットはどの程度まで同じ変化をするか

TABLE VII: T-test results for GitHub revisions and Stack Overflow edits.

Sample	Q1		Q2		Q3		Q4	
	μ	p	μ	p	μ	p	μ	p
Control	779.94	–	602.01	–	490.40	–	379.43	–
Simulated 1	779.94	0.99	602.09	0.99	496.67	0.71	381.34	0.88
Simulated 2	779.94	0.99	600.98	0.96	501.13	0.54	379.80	0.97

進化のタイムラインは重なっているものの、GHスニペットと、それに対応するSOスニペットは互いに独立して進化している

結論

- SOとGH上のスニペットの変更履歴を調査
- SOでは76.3%, GHでは22%のスニペットが進化している
- コンテンツの作成者がスニペットの進化に重要な役割を果たしている
- スニペットは各プラットフォームで独立して進化している

所感

- SOの投稿がどれだけ利用されているかを見ることで、投稿の品質を予測できるのではないかと感じた
- SOとGHで進化の割合が違うのは、なにが原因なのか気になった

補足

データセット作成について

コミットからコードコンテキストを抽出

```
1. // Code
2. // Code
3. initPlayback($elem, $wrapper, moveTemplate, opts.autoPlay, opts.still)
4. updateNavi($navi, 0);
5. updateButtons($elem, 0, amount);
6. if(opts.resize) updateHeight($slideContainer, $slides, pos);
7. + disableSelection($elem)
8. function moveTemplate(indexCb, animCb) {
9. ....
10. - return function () {
11. ... 1.
12. ...
13. ..
14. ..
15. + function disableSelection($e) {
16. + // http://stackoverflow.com/questions/2700000/how-to-disable-text-selection-using-jquery
17. + return $.each(function() {
18. + $(this).attr('unselectable', 'on').css({
19. + 'moz-user-select': 'none',
20. + 'webkit-user-select': 'none',
21. + 'user-select': 'none'
22. + }).each(function() {
23. + this.onselectstart = function() { return false; };
24. + });
25. + });
26. + };
27. +
```

Foo.py
r₁, Origin Commit, C₀

```
1. // Code
2. // Code
3. initPlayback($elem, $wrapper, moveTemplate, opts.autoPlay, opts.still)
4. updateNavi($navi, 0);
5. updateButtons($elem, 0, amount);
6. - disableSelection($elem)
7. + if(opts.resize) {
8. + disableSelection($elem)
9. + }
10. function moveTemplate(indexCb, animCb) {
11. ....
12. ...
13. ..
14. function disableSelection($e) {
15. // http://stackoverflow.com/questions/2700000/how-to-disable-text-selection-using-jquery
16. return $.each(function() {
17. $(this).attr('unselectable', 'on').css({
18. 'moz-user-select': 'none',
19. 'webkit-user-select': 'none',
20. 'user-select': 'none'
21. }).each(function() {
22. this.onselectstart = function() { return false; };
23. });
24. });
25. };
26.
27.
```

Foo.py
r₁, First revision Commit, C₁

Code Context Lines Added Lines Deleted

コミットからコードコンテキストを抽出

- ・ファイルの履歴からすべてのコミットを抽出後、SO投稿への参照が導入されたコミットを探す
- ・コードコンテキストは、次のリビジョンの緑と青の行で構成される
(既存の行に導入された行を追加する)

Code Miner

- git プロジェクトのクローンを作成し、コードコンテキストの概念を使用して、ファイル履歴をマイニングするツール
- SOの投稿を参照しているGHプロジェクトはPostReferenceGHテーブルに格納され、次のようなURLになる
(https://raw.githubusercontent.com/RationalAsh/freelMU/master/debug/decode_float.py)

Code Miner

TABLE III: Deriving GitHub project information.

Element	Example
HEADER	<i>https://raw.githubusercontent.com</i>
USER/REPO-NAME	<i>RationalAsh/freeIMU/</i>
BRANCH	<i>master</i>
FILE-PATH	<i>debug/decode_float.py</i>

- ・URLをいくつかの要素に切り分け, そのなかからプロジェクト名を得る (USER/REPO-NAME)

Code Miner

- 次にGHのAPIを用いて, プロジェクトの情報を収集
- そして, リポジトリをクローンし, 再利用されたコードスニペットを含むファイルを探す
- 特定したファイルに対してgit blameを実行
(指定したソースファイルのコミットグラフを作成)
- 対象となるファイルの変更点とともに,
コミットのメタデータ全体を抽出