FINAL PROJECT REPORT

COURSE : SOFTWARE AND DATA ENGINEERING

TOPIC

Impact of stack overflow code snippets on software cohesion: a preliminary study

By

ABHISHEK RAGHAV (B19CSE004)

SHANTANU BAKSHI (B19CSE019)

Objective

Determination of the effect of copying code from stack overflow snippets in production code by using cohesion metrics such as low level similarity based class cohesion and class cohesion to check the final quality of the code.

Introduction

“Good coders borrow, great coders steal”

Copying code from Stack Overflow is a form of code cloning; that is, duplicating code from within a project or between projects and reusing it. Depending on who you ask, as little as 5-10% or as much as 7-23% of code is cloned from somewhere else.

Regardless of the exact amount, code cloning is extremely common. Boilerplate code is essentially code repeated regularly throughout a project. Chances are pretty good that those coders aren’t typing each of those by hand.

The goal of the research is to investigate whether the copied snippet contributes to code quality, or disimproves it.

The metrics used for the following are LSCCC and CC both variations of the class cohesion metrics .

Methodology

We have chosen the SOTorrent database for this project which is available on Google BigQuery. We have first of all migrated the database from BigQuery to CSV and then chosen randomly 500 samples from the PostReferenceGH table, all of which are in Java. We then evaluated the LSCC and CC metrics for all the classes as described further.

What actually are the metrics ?

In computer programming, **cohesion** refers to the *degree to which the elements inside a module belong together*. In one sense, it is a measure of the strength of relationship between the methods and data of a class and some unifying purpose or concept served by that class. In another sense, it is a measure of the strength of the relationship between the class's methods and data themselves.

Class cohesion (CC)

Now what the class cohesions metric checks is the similarity ratio of similarities between all possible methods in a class and is given by

$$CC_c = 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\frac{|I_i \cap I_j|}{|I_i \cup I_j|}/k(k-1)$$

Where i represents the set of attributes referenced by method i, while k represents the total number of methods in a class.

Low-level Similarity Class cohesion (CC)

This metric does the job of finding the ratio of summation of common attributes accessed by a method and is given by

$$LSCC_c = \begin{cases} 0 & if l = 0 \ and \ k > l, \\ 1 & if (l > 0 \ and \ k = 0) \ or \ k = 1, \\ \sum_{i=1}^{l} x^i(x^i - 1)/lk(k-1) & otherwise. \end{cases}$$

Where k and l indicate the number of methods and attributes of a class respectively. Many attributes and single method makes this metric very cohesive .

Dataset and preprocessing :

The database used is the SOTorrent dataset which provides the version history of individual posts and links from SO posts to other datasets like GitHub. It contains a total of 1314056 records consisting of snippets related to java , R, python etc.

In order to get quantifiable insights we consider two scenarios : version , just prior to the addition of the snippet, **pre-SO snapshot** and the one just after the addition of the snippet, **post-SO snapshot.**

History of commits is used to measure the variance in cohesion . the snippets which were ignored were the ones with direct pushes, because there was no pre-SO snapshot of them .

Important questions :

Starting with an end in mind we have two principal questions that we need to answer

1. Quality difference .
2. Quality evolution .

Previous research

Existing work consists of research studies that have been carried out on different aspects of Stack Overflow, examining the quality of code available on SO and GitHub repositories from different perspectives. For example, research by Kavaler et al used view counts and user scores to assess the quality of SO answers related to new Android APIs, finding that questions posted with incentives .In another research quality of SO code snippets was investigated by verifying the misuse of API calls. They found that approximately 31% of the analyzed SO posts possibly incorporate program shattering API usage contraventions

Comparing Existing work and Original work:

QUESTION 1 : quality difference

The LSCCC and CC metrics lie between 0 and 1 . The changes in LSCCC and CC metrics are observed for pre and post -stack overflow copy snaps of the classes where the method was copied .

2 cases were observed namely varying and constant . where positive variation occurs the cohesion has gone up significantly after copying from stackoverflow and vice versa for negative variation .

The variation is further of 2 types

1. deteriorating : When the metric value changes and the value decreases,
2. improving. When the changing value is increased,

Existing results :

The existing results are as follows

(LSCC) remains constant in 39% of cases and varies in the remaining 61% of cases. Out of the 229 cases that vary, 69% show deterioration in cohesion, as against the 31% that show improvement in LSCC.

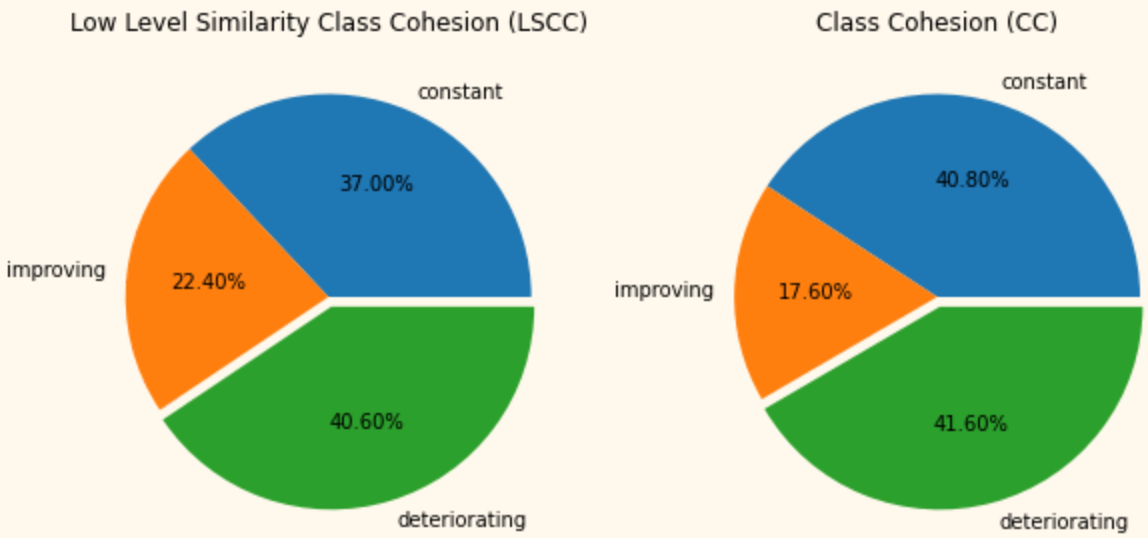
Also in cases where copied code does not affect metrics the effect on the quality is affected negatively. This indicates that the cohesivity is adversely affected by external sources.

Obtained (orignal) results :

We observed in our results that using the LSCC metric, **40.60%** of classes deteriorated after copying code from StackOverflow. The results remained constant in **37.00%** of classes whereas they improved in solely **22.40%** of the classes.

A similar pattern was observed in the CC metric, where **41.60%** of classes deteriorated, **17.60%** improved and **40.80%** remained constant.

The figures can be verified from the pie charts obtained by us during our analysis.



QUESTION 2 : quality evolution

Now we wish to check if the effect of copying the code is more permanent or dissipates after a certain extent. Some cases were checked **for deterioration** by examining the commits after the Stackoverflow snapshot .

This checks whether the class is able to recover from the change in cohesion and regain normalcy . Three cases are possible a) recovered b) not recovered c) partial recovery .

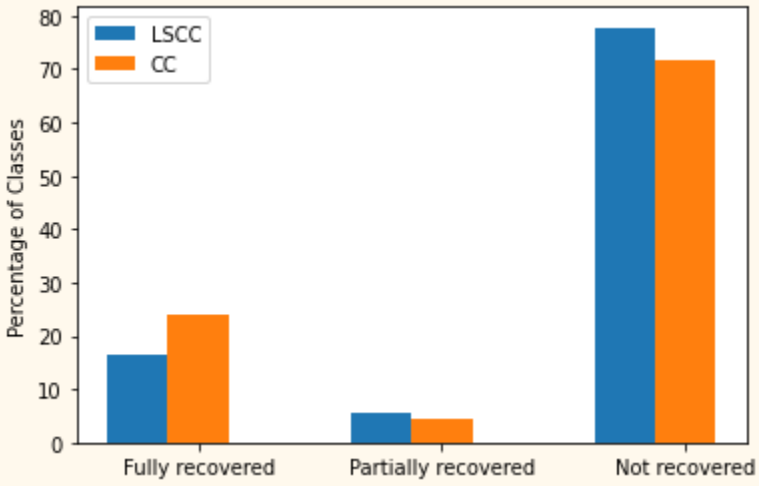
Existing results :

They obtained that lsccc metric for 78% percent of classes couldnt recover their pre snapshot cohesion value and 18% recovered and 4% are in the middle

The same distribution for CC is 70% 25% and 5%

Obtained (orignal) results :

In this case too, the results obtained were close to the original results. After observing the patterns for classes if they were able to fully recover from cohesion loss, partially able to recover (i.e. close to 50%) or not at all, the following results were obtained:



We observe that using the LSCC metric, **16.60%** were able to fully recover, **5.60%** to partially recover and a staggering **77.80%** of classes could not recover at all.

Similarly, in the case of CC metric, **23.80%** were able to fully recover, **4.40%** to partially recover and **71.80%** could not recover from the loss of cohesion.

In the following graphs, we can also have a look at the cohesion of a file over a period of time. Three samples are shown here, one each for fully recovered, partially recovered and unable to recover.

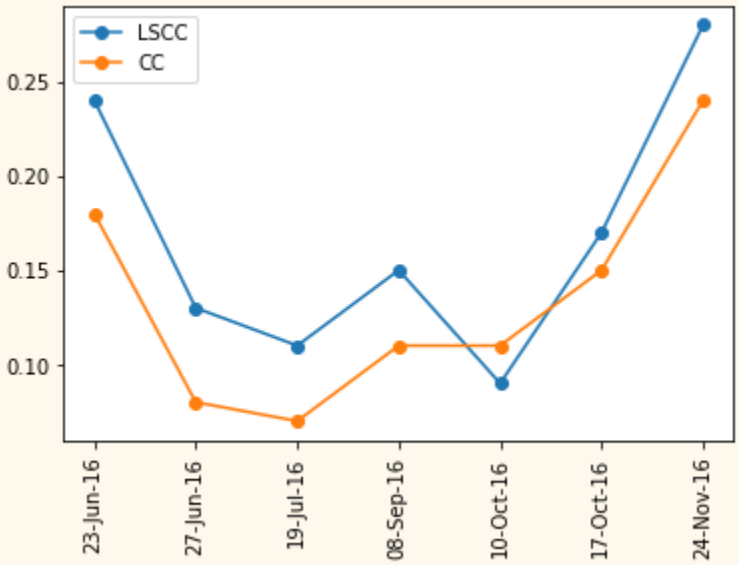


Fig: Cohesion history of fully recovered class

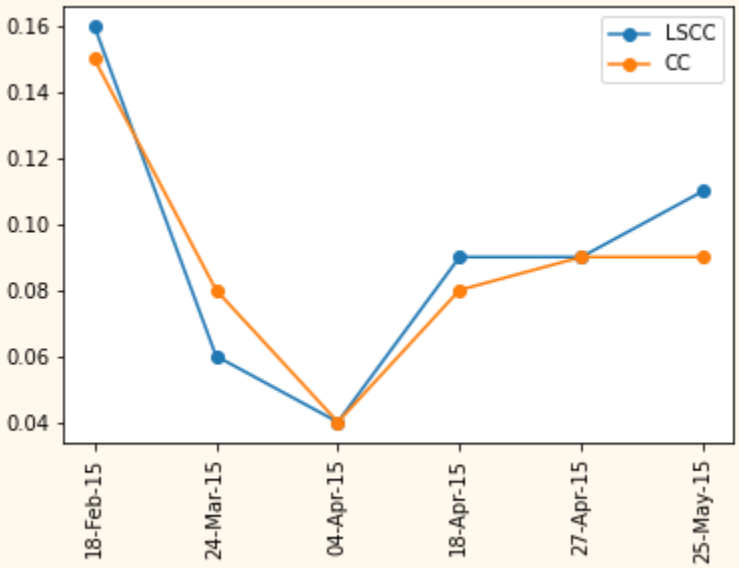


Fig: Cohesion history of partially recovered class

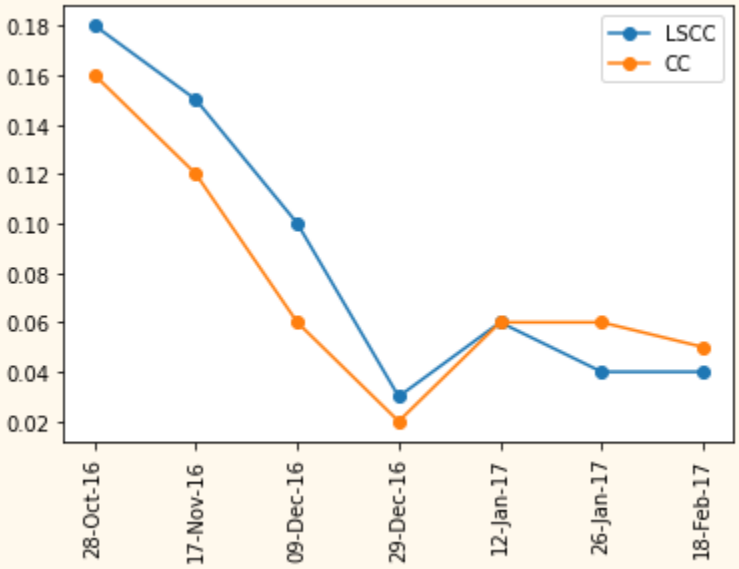


Fig: Cohesion history of not recovered class

Conclusions :

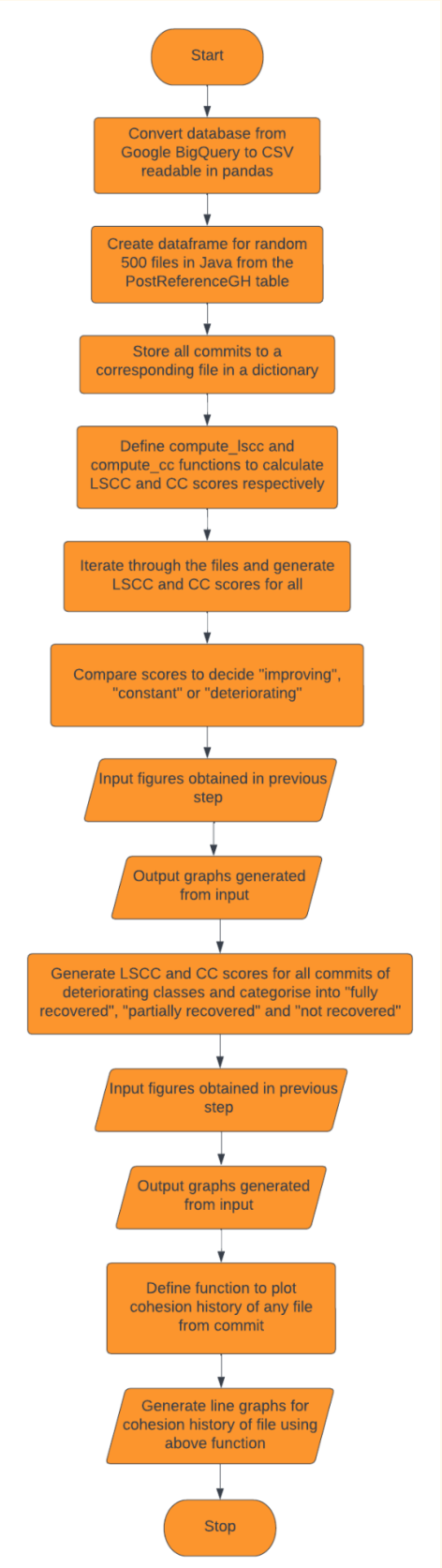
According to the results shown we see that in more than 70% cases the data fails to regain its cohesiveness once assimilating the stack overflow snippet and out of these the majority of cases

experience a loss in cohesiveness in general after some commits (which is called the evolution).

Specifically , in a sample of 500 classes almost 42 percent of the cases show a deterioration in the cohesiveness and more than 70 percent of these show a deterioration in quality of code cohesivity as seen after subsequent commits .

One thing that could explain this is the hasty copying of code could be caused by external factors for example, time pressure on the developers which would lead to incorrect adding of code into the codebase . Major assumption made during this research is regarding the effectiveness of metrics and their correlation to the quality of the code.

Schematic Diagram:



GitHub Link:

https://github.com/imraghav20/SDE-Course-Project

YouTube Link:

https://youtu.be/ZlJZJtI2LKE