Vote Variance Factors Analysis of Similar Stack Overflow Posts (Q&A)

A study on Clone Detection within the Stack Overflow Code Snippets

Md Manik Hossain
Computer Science and Software Engineering
Montreal, Canada
Student Id: 40108891
E-mail: manik.ruet08@gmail.com

Full Project Link Altogether: GitHub.com[1]

Abstract—Duplicate codes from online sources are code fragments which are copied from the same online sources such as Stack Overflow. Stack Overflow is a popular online Q&A website where developers discuss coding problems and share code examples for seeking and providing answers. Due to the popularity of this technical questions and answering website, developers are frequently re-using the code snippets for asking questions and answering the questions within the Stack Overflow. In this paper, we conduct a case study only with java posts (i.e., questions and answers) on Stack Overflow, to investigate i) similar Q&A in terms of cloned code snippets ii) vote differences among the cloned Q&A iii) accountable factors that affect vote differences between cloned Q&A through mixed modeling. For cloned questions, we found 20% (i.e., 60,079) java questions that are cloned to each other from 291,427 java questions. In terms of cloned answers, we found 12.24% (i.e., 16,316) java accepted answers that are cloned to each other from 133,243 java accepted answers. Furthermore, we collect around 45 potential factors for both questions and answers; build mixed-model to finding out the most responsible factors that contribute more to the vote differences. Finally, we found 25 responsible factors that affected vote differences of those cloned questions and answers. So, this study reveals that developers commonly use Stack Overflow for posting Q&A and use code snippets repeatedly as an example, but before voting users consider 25 factors that make vote differences among cloned questions or answers.

Index Terms—clone detection, mixed modeling, vote differences, NiCad, Stack Overflow

I. INTRODUCTION

Technical Q&A websites such as Stack Overflow gained popularity because of sharing knowledge and solving programming issues over the internet by the programmers. When developers stuck with any programming problem they ask for help from other expert developers by posting an example code snippets through the most popular technical Q&A website, Stack Overflow. In this way, SO accumulated around 16.5 million questions, 26 million answers, 9 million accepted answer, and 10 million users by the end of 2018. From previous study, we have seen that developers copy and paste the code snippets from or to Stack Overflow when they face any programming problems or developers post answers in order to solve the problem by providing their own project or other resources code snippets as an example [1,2]. Le An et al. found 232 code snippets have been reused from Stack Overflow in 62 Android apps, and 1,226 Stack Overflow posts that contain example code snippets which are cloned from 68 Android apps, suggesting that developers may have copied the code of these apps to answer Stack Overflow questions [5] or copied from Stack Overflow to android apps for development.

Although the code copying and pasting trend have been running continuously way over the years on the Stack Overflow community, it accumulated an impressive knowledge base on snippets of code that are nicely documented on Stack overflow [3]. In addition, as a measure of its popularity, SO received more than half a billion views on the first 30 days of 2017 alone [4].

Most of the times developers search for answers of a specific programming related problem and they may find answers in the stack overflow website but that answers could be problematic or did not work for his existing problem, hence, developer may re-post a question with different title, context and same example code snippets for seeking an accurate answers from the SO developers with that example code snippets or from their own projects code snippets. Answerers also provide similar types of answers with an example code snippet for multiple times for multiple questions. Therefore, developers also post answers by searching online tutorials or from their project as an example code snippets to solve the problems or they collect answers from inside the stack overflow which are related to the similar problems to gain reputation points. By following this approach, developers are generating huge amounts of cloned code snippets of questions or answers over the years.

Each questions or accepted answers have respective votes that added to the reputation scores of an asker and answerer account as well. Although several questions and answers are similar in terms of their code snippets, they have different votes that provided by the SO users after considering various

[1] https://github.com/MMH08/SOEN-7481-Research-Project

types of factors which affect the vote differences.

From this stage, we wish to investigate what are the factors that cost vote differences of similar questions or answers and hence, we build mixed-model for finding out those responsible factors that contribute more to the votes for this differences between the clone pairs (cloned code snippets).

In this paper, we conduct a quantitative study to investigate similar questions and answers in the technical Q&A website, Stack Overflow. In terms of questions, we analyze 910, 936 (0.91 M) java questions that have accepted answers and have at least 10 LOC (i.e., lines of code) after filtering from total 1,709,389 (1.7 M) Java question that have accepted answers. From 0.91M java questions, we extracted 1,315,097 (1.3 M) code snippets for clone detection and further analysis. In terms of answers, we analyze 517,482 (0.51 M) java accepted answer that have have at least 10 LOC (i.e., lines of code) after filtering from total 1,707,497 (1.7 M) java accepted answers. From 0.51M java accepted answers, we extracted 656,180 (0.65 M) code snippets for clone detection and further analysis. We use a state-of-the-art clone detection tool [6] NiCad [7], to identify duplicate code within the Stack Overflow code snippets for both questions and answers in terms of block and functions. To ensure that code clones reported by NiCad are real code clones, we manually and randomly validate several clone groups found in both questions and answers in the Stack Overflow.

We answer the following two research questions:

- **RQ1: What are the factors that affects the vote differences of each cloned Questions (i.e., cloned in terms of code snippets)?**
  For questions, from 1,315,097 (1.3 M) code snippets of java questions which have accepted answers, we got 2,185 (2.1 K) clone groups from 8,865 (8.8K) unique questions for blocks and 16,516 (16.5 K) clone groups from 51,214 (51K) unique questions for functions on Stack Overflow. We found 20% code snippets are cloned to each other from 0.37M extracted code snippets of java questions (0.91M) that have accepted answers and at least 10 Lines of code. This result provides a quantitative evidence of potential code snippets copying and pasting for posting or asking questions with similar example code snippets within the Stack Overflow technical websites by the users over time from 2008 to 2018. Afterward, we will build mixed model to find the more responsible factors that contribute on the vote differences for questions on Stack Overflow.

- **RQ2: What are the factors that affects the votes differences of each cloned Answers (i.e., cloned in terms of code snippets)?**
  For answers, from 656,180 (0.65 M) code snippets of java accepted answers, we got 8,52 clone groups from 5,517 (5.5K) unique accepted answers for blocks and 2,804 (2.8K) clone classes from 10,799 (10.7K) unique accepted answers for functions on Stack Overflow. We found 12.24% code snippets are cloned to each other

from 0.13M extracted code snippets of java accepted answers. This result also provides a quantitative evidence of potential code copying and pasting within the Stack Overflow for answering questions using similar kind of example code snippets by the users over time from 2008 to 2018. Afterward, we will build mixed model to find the more responsible factors that contribute on the vote differences for accepted answers on Stack Overflow.

II. CASE STUDY DESIGN

In this section, we describe the data collection, data filtering, data extracting and clone detection approach using NiCad clone detection tool and afterward, we will use the clone detection results to answer our two research questions. Figure-1 shows a general overview of our data mining and data processing approach. We will describe each steps in our data processing approach in this Section-II. The corresponding study results, data and all other scripts are available on GitHub [2].

A. Data Collection : code snippets corpus

In order to set up our environment for study, we obtained the Stack Overflow data dump (published in Dec 2018) in XML format and download the data dump from SO official website [3] in XML format. Then we imported the XML data into MSSQL for further analysis or mining the SO data. The data dump contained 42,736,511 posts, including 16,635,796 questions and 26,003,926 answers until 2018.To perform our case study, we extract Java questions and answers from Stack Overflow . Each discussion on Stack Overflow includes a question and zero or more answer posts and their meta data (e.g., body, creation date and number of votes). Each questions are typically tagged with maximum five terms describing the categories under which these Q&A discussions are grouped such as Java, C#, .NET and so on.

B. Data Filtering Approach of Questions and Answers

In this next mining step, we extract all the relevant posts for our study context using simple or advanced SQL query to MSSQL [4]. Since we focus on Java questions and answers, we extracted all the java related posts (i.e., questions and answers) that are associated with "Java" tag (3,229,726 questions). We then further filtered these discussions to ensure that they also have accepted answers for questions with SQL. As we are only interested in finding similar questions and answer in terms of code snippets from Stack Overflow, we further filtered our data-set by SQL query after considering those posts that contain code snippets as an example code in the posts. For our analysis, we focused on the source code of java questions and answers of the Stack Overflow posts and if the code snippets are clone, we consider both questions or answers are similar. From this stage, we further filtered the questions that have at

---

TABLE I
QUESTIONS: POSTS FILLETING APPROACH OF JAVA QUESTIONS

# Steps of Questions Filtering Approach	Total	Percentage (%)
All Java Questions have Accepted Answer	1709389 (1.7 M)	100%
All Java Questions with Code Snippets	1492881 (1.5 M)	87.33%
All Java Questions have at least 10 LOC	910936 (0.91 M)	53.29%
# Extracted code snippets from 0.91 M Questions	1315097 (1.3 M)	-

TABLE II
ANSWERS: POSTS FILLETING APPROACH OF JAVA ACCEPTED ANSWERS

# Steps of Accepted. Answers Filtering Approach	Total	Percentage (%)
All Java Accepted Answers	1707497 (1.7 M)	100%
All java Accepted Answers with code snippets	1419125 (1.4 M)	83.11%
All Java Accepted Answers have at least 10 LOC	517482 (0.51 M)	30.30%
# Extracted code snippets from 0.51 M accepted answers	656180 (0.65 M)	-

least 10 LOC (lines of code) after developing our first java code snippets extractor tool (ToolOne) [5] Therefore, finally we got 910,936 java posts that have accepted answers along with example code snippets contain at least 10 LOC.

Since prior work suggested the use of highly voted code snippets, we only focused on code snippets in questions that have accepted answers and answers those are java accepted answers because that have a high number of votes provided by the Stack Overflow users [8]. We wish to study only the posts which contained code snippets with at least 10 lines of code because smaller code blocks less than 10 LOC would have too many false positives. This due to the fact that in less than 10 lines of code users may share a simple one line of variable, if or for loop statement or a few lines of block of code that may flag common for all the example code snippets. Therefore, selecting 10 lines of code restricted the size of our datasets and it also decrease the false positive (FP) rate by the clone detection tool. Finally, we decided to keep 10 LOC as the threshold of clone detection tool and collected 1315097 (1.3 M) code snippets from 910936 (0.91 M) java questions. We also followed the same approach for answers that have been accepted by the askers (i.e., owner of the question). As the approach for both questions and answers is the same we will not discuss the same story for the accepted answers. Therefore, in terms of questions, Table-I summarizes how we filter questions and get the final number of posts after finishing the filtering. In terms of accepted answers, Table-II summarizes how we filter accepted answers and get the final number of posts after finishing the filtering.

C. Pre-Processing and Extracting Code Snippets from Stack Overflow

Using SQL query we export all the identified posts into CSV file for further processing by java ToolOne which extract each code snippets into separated java files (i.e., .java). Therefore, after finishing the selection of SO java posts and exporting of all the Stack Overflow posts, we followed a straightforward

[5]https://github.com/MMH08/Tool1-Extraction-of-SO-JAVA-CodeSnippets

and traditional techniques to identify the code snippets inside each Stack Overflow posts. Actually, the whole body of a questions or answers are comprised of the HTML codes and tags. Hence, in Stack Overflow, the posts table store the body of a post in HTML format. Code elements in these posts are surrounded by the <code>tag. Therefore, we identify the code segments by searching for the <code>tags after developing ToolOne which have been developed to extract code snippets from tag based HTML code for this research purposes. After extracting the code snippets from HTML format text, we created java files (.java) for each code snippet using our developed java tool. We randomly and manually investigate several java files and the blocks of code to check false positive in terms of blocks and function of the text between these <code>....</code >tags that contain the code snippets. Although we filter java questions and answers, we noticed that posters usually not only use java code snippets in a questions or answers as an example code snippets but also they add other languages (e.g., XML, C#, JavaScript, SQL, stack traces, plain text, URLS and so on) as a supporting and additional information during asking for help or to solve the problems. Therefore, it was nearly impossible to remove all those irrelevant code snippets from the same post (i.e., questions or answers) that are not related to java from the code snippets of HTML body. As a result, we noticed that our error rate of parsing code snippets using NiCad was higher and we noticed that the error rate (i.e., 68% for questions and 77% for answers) of the code snippets of questions was lower than the error rate of the code snippets of answers.

During manual study of code snippets, we observed some problematic code on Stack Overflow that create challenges for us to identify and separate all programming languages code snippets by our ToolOne. But few errors in the java code snippets were easy to solve by filtering and replacing information by if else conditions in the ToolOne and we were successful to solve some many of them (i.e. problematic code snippets) using our ToolOne to ready for parsing on NiCad. Because problematic code would not parse by NiCad clone detector and throw errors and skips those errotic code blocks
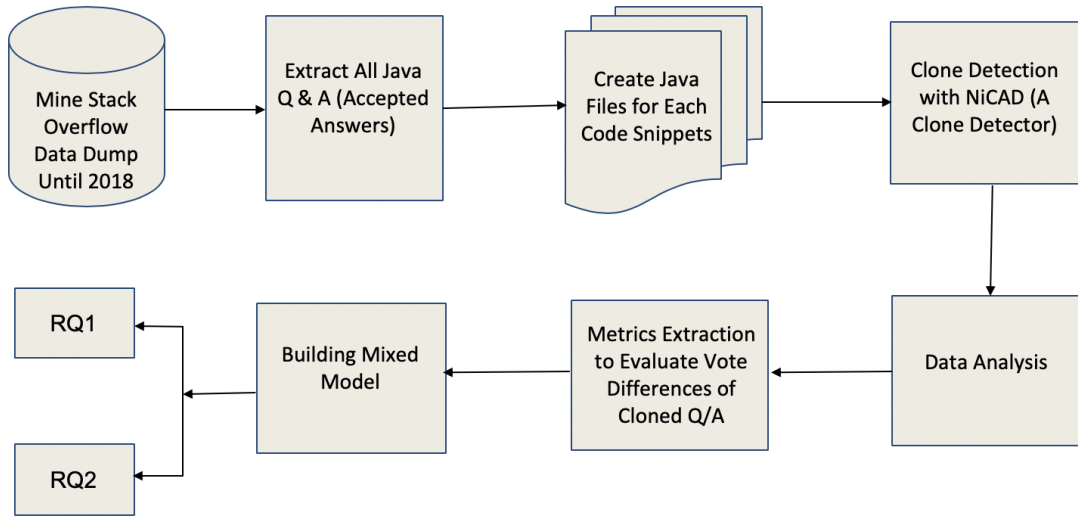
Fig. 1. Overview of our data processing approach

from the extracted data corpus. If the detection tools skips these problematic code for minor issue on the coding, we will lose a huge amount of code snippets from our data corpus. Therefore, we manually try to remove those minor issues such as extra text in coding as comment, missing brackets or semicolons, and other unnecessary information in code that creates error while parsing the code and throw exception by the parser of NiCad, a clone detector. These errors are as follows with an example Stack Overflow post that associated with postid - 36731552 [6]:

- **Message or Comment in Code Blocks:** SO users sometimes provide message in the code section without using double slash "//" for putting comments which leads to parsing error by the clone detector. This example post is an ideal combination of these type of problematic code and this example post (i.e., questions) is associated with post Id : 36731552.
- **Missing Semicolon(;):** Some users provided code snippets with problems in the code like they forgot to add a semicolon at the end of the statement that leads to parsing error in clode detector.
- **Missing Brackets or Extra Brackets** (}): Sometimes users also do not focus on the closing brackets or they put extra brackets while posting code snippets as an example on SO while asking and answering questions.
- **Others:** There are other such kind of error we observe like calling functions without the function definitions in the code blocks, use "..." in the middle of the code snippets and many more. Basically, we found a lot of different types of erroneous coding format in the Stack Overflow code snippets.

In order to reduce error rate during clone detections using NiCad, we separate all the code snippets in terms of blocks

[6]https://stackoverflow.com/questions/36731552/recursive-algorithm-with-depth-first-search

and functions. Because at present NiCad supports two granularities, functions and blocks, and five languages, C, C#, Java, Python and WSDL [7]. Therefore, we only extract code either in block (i.e., consist of one or more declarations and statements) or functions (i.e., that contains function body in the code snippets) using our code snippet extractor java ToolOne. So, in order to avoid huge amount of errors and mixing of both (i.e., blocks and functions) we separated code snippets for accurate clone detection to find out the similarity of questions and answers in separate way. This is due to the fact that NiCad can not parse block of code during function granularity as well as NiCad can not parse functions of code during block granularity during clone detection. This approach of separating code snippets in terms of blocks and functions made our clone detection more easier.

However, after following this ideal approach, the error rate of the clone detection using NiCad is still high which accounted 68% for questions and 77% for the answers from 1.3 M and 0.65 M code snippets respectively. This is because of the problematic code on Stack Overflow posted by users and other languages code that askers usually add for supporting their questions or answers as an example. So, the successful parsing rate of the extracted code snippets for both questions is 32[Todo: Add table for show error rate and how many cloned have found from]

D. Finding similar Q&A by code snippets using clone detection

Once we extracted all the java code snippets, we planned to detect the duplicate code snippets within the stack overflow in terms of blocks and functions granularity and separately for questions and answers. For detecting the clone we use NiCad [7] clone detection tool. NiCad is a token based clone detection tool developed by J. R. Cordy et al. from Saskatchewan University [7]. NiCad can detect three different types of clones such as Type-1 which is exactly similar code

snippets, Type-2 that is syntactically similar code snippets, and Type-3 which is copied code with further modifications [9]. NiCad also can detect clone of three different languages such as Java, C++ and Python. We choose NiCad clone detection tool because of better performance in comparison with other 10 clone detectors reported by Svajlenko et al. [6] . He also examined that NiCad achieved higher precision and recall, in comparison to the other 10 clone detection tools [6]. Therefore, we pass all the code snippets through NiCad separately for questions and answers in terms of blocks and functions. As we mentioned that our code parsing error rate is higher, hence NiCad automatically filter out huge amount of irrelevant code that are not corresponding to the syntax of Java and processed further only with those who parsed successfully by parser. Basically, we did not change the default settings and processed the clone detection with the default setting of NiCad, i.e., each clone pair has equal or more than 70% of similarity and the clones contain at least 10 LOC (lines of code).

III. CASE STUDY RESULT

In this section, we will discuss about the clone detection result generated by NiCad and how we extract the results from HTML file for more processing the extracted dataset. As we discussed in Section-II-C, we filtered the data from Stack Overflow and pre-processed the code snippets for cloned detection using NiCad clone detector. Afterward, once the code snippets passed through the NiCad clone detection tool, it generates the result in terms of block and functions in HTML file formats and we run NiCad separately for questions and answers. In terms of questions, we found 20% code snippets are cloned to each other and this cloned code snippets comes from 60,079 (60K) java unique questions of Stack Overflow. In other words, 60,079 java questions (20%) are similar (i.e., cloned) to each other in terms of their code snippets. Similarly, in terms of accepted answers, we found 12.24% code snippets are cloned to each other and this cloned code snippets comes from 16,316 (16.3K) java unique accepted answers of Stack Overflow. In other words, 16,316 java accepted answers (12.24%) are similar (i.e., cloned) to each other in terms of their code snippets. Table-III show our clone detection results for both questions and answers with fail and pass rate (e.g., raw number and percentage) produced by NiCad during the clone detection processing.

A. Extracting Clone Detection Results for Further Processing

As we mentioned in section III NiCad generate the clone detection result in the HTML format, we need to again extract the cloning results for further analysis. In order to extract the cloning result from HTML format, we develop one more tool (ToolTwo [7]) which find the post id along with clone groups, clone size in terms of line numbers, similarity between clone in percentage, creation date of the post and granularity (i.e., blocks or functions). In addition, this ToolTwo extract all the aforementioned factors from the HTML file and store into the

CSV file for further analysis. Figure-2 show the different part of the results of clone detection that generated by NiCad. We import these CSV formatted cloned data-sets into the MSSQL server for understanding the vote differences of each cloned groups after finding the scores of all the post from main data corpus (i.e., Stack Overflow database). Because we considered that two posts (i.e., question or answer) are duplicate or similar, if they have cloned code snippets. Therefore, we need to find post information that associated with the cloned code snippets and also find other additional information (e.g.,, votes, creation date, post owner id) from SO database using SQL joins of those similar questions and answers.

IV. PRELIMINARY ANALYSIS OF CLONE DETECTION RESULT

In this Section-IV, we will discuss about measuring and analyzing vote differences of the cloned classes generated by NiCad clone detector. In addition, how we normalize the vote variances value for better understanding the data distribution by boxplot visualization.

A. Calculating vote differences of cloned groups

Once we extracted the clone detection results from the HTML formatted file generated by NiCad, we wish to investigate the vote differences of each cloned groups. As we mentioned in Section-III-A that if two posts (i.e., questions or answers) are similar to each other in terms of cloned code snippets, we need to find out the vote differences among the group members. This is due to the fact that we want to find out the responsible factors that affect their (i.e., similar questions or answers) vote difference. Therefore, investigating vote differences is inevitable because without any existence of vote variance between the group members we can not proceed with those cloned data. So, we need to consider only those cloned groups which have the vote variances among the group members.

To understand the vote differences, we calculate the several terms (e.g., 25 percentile, max , min , mean, median, variance of scores) in R studio after importing the cloned CSV data-sets. We calculate the vote difference for each group from cloned data-sets for both questions and answers. The R scripts of calculating those terms are available on GitHub [8] inside the R project. To see the vote differences, we mostly focus on the variance because variance show the actual scenario of the difference of vote among the group members. For better understanding the differences after calculating the variance we visualize the result in boxplots because boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles and this graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across the groups by drawing boxplots for each of groups.

From boxplots we mainly focus on the zero variance to know that how many group are there they do not have any

---

[7]https://github.com/MMH08/Tool2-ParseHTML-Results-Of-NiCad

[8]https://github.com/MMH08/R-Mixed-Model-Cloned-Q-A

TABLE III
OVERVIEW OF CLONE DETECTION RESULT BY NICAD

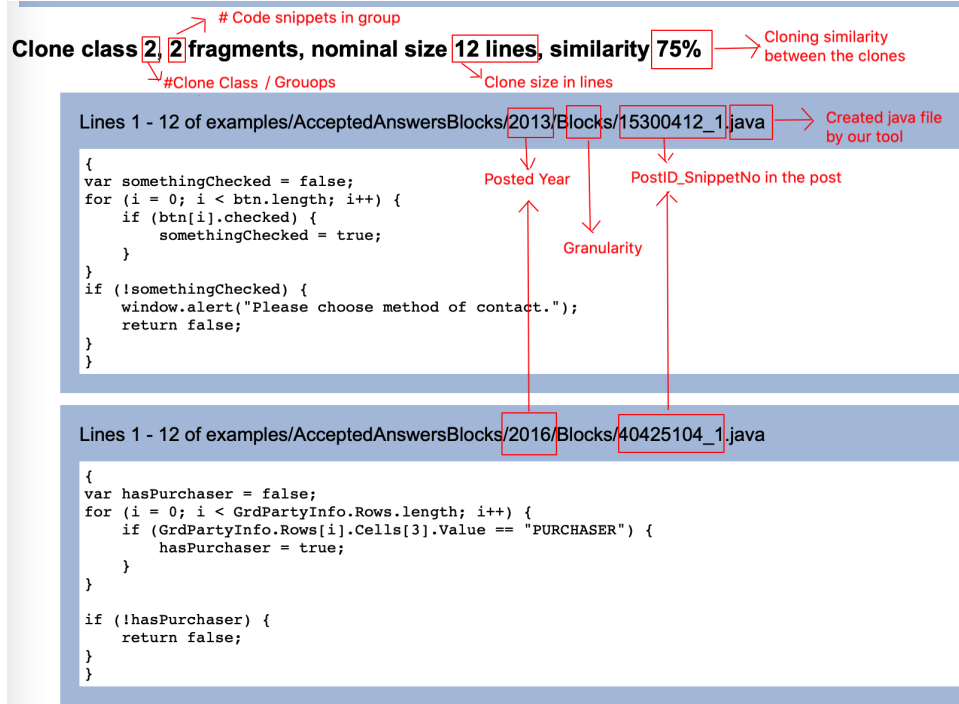| Post Type	#Post	#Code Snippets	Fail (%)	Pass(%)	#Clones of Blocks	#Clones of Functions	#Unique Posts of all Clones
Questions	910936 (0.91 M)	1315097 (1.3 M)	0.93 M (68%)	0.37 M (32%)	2,185 (2.1 K)	16,516 (16.5 K)	60,079 (60K)
Answers	517482 (0.51 M)	656180 (0.65 M)	0.52 M (77%)	0.13 M (23%)	852	2,804 (2.8K)	16,316 (16.3K)

Fig. 2. Manual labeling of different parts or factors of HTML formatted clone detection results generated by NiCad clone detection tool.

vote differences among the group members. But the most interesting and challenging part was few cloned groups generate variance in big number such as 1000000, 1500000, hence it was nearly impossible to identify vote variances using this traditional way (i.e., directly use built-in var() function of R studio). Because all the vote variances represents a horizontal straight line in the figure that actually does not tell us anything. Figure-3 represents the real issue about the problem that we faced and we manually labeled few parts of the figure for better understanding the issue easily. In addition, we observed that there are around 50% of groups that have zero vote variance from all the cloned groups. As keeping those unnecessary data (i.e., zero variance groups) on the entire cloned data-sets would create bias results in modeling, we removed those data from our study to achieve the better result from mixed modeling.

B. Apply Shannon Normalized Entropy (SNE) for better visualization

To solve this problem we normalize the vote variances of each groups by applying Shannon Normalize Entropy (SNE). The formula of SNE can be written like Figure-4. We implement this formula in R studio and calculate the vote variance with all the cloned groups datasets and this time we found more clear scenario that helps us to understand the differences
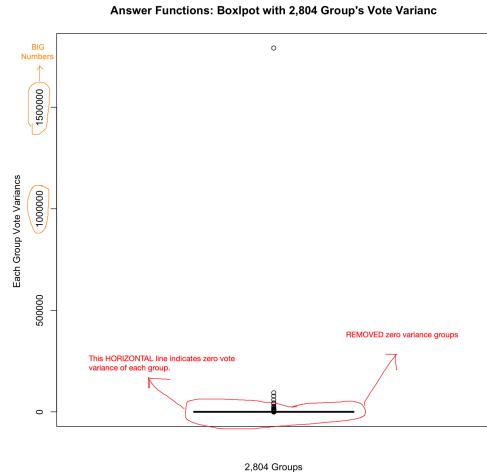

Fig. 3. Boxplots with BIG numbers create problem to visualize smaller numbers in the figures for understanding the data distributions.

easily. Finally, we draw boxplot with the normalized variance and SNE normalize the vote variances within the limit 0-1 that represents in Y-axis in the Figure-5. Therefore, this figure indicates better visualization of vote variance compared to

$$H_n(p) = -\sum_i \frac{p_i \log_b p_i}{\log_b n}.$$

Fig. 4. Shannon Normalized Entropy for calculating the vote variances of each cloned groups.
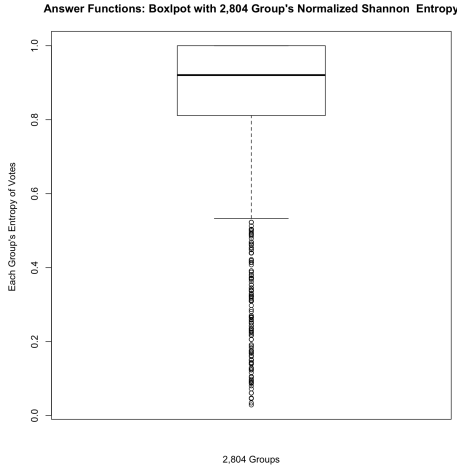

Fig. 5. In order to know the vote differences among the group member of the cloned questions or answers, we applied Shannon Normalized Entropy (SNE). SNE helps to demonstrate better visualization in vote differences of each cloned groups.

Figure-3.

V. FACTOR COLLECTIONS AND POPULATE FACTORS DATA

In this Section-V, we will discuss how we collect the potential factors affecting the vote differences of similar questions or answers in terms of their code snippets. Once we collect all the potential factors, we will populate all the factors data. We follow 3 approaches for populating all the factor's data such as development third java tool, using MSSQL server and R Studio.

As we mentioned in Section-I that we will build mixed model to find out what are the factors that cost vote differences among the similar questions or answers. Therefore, in order to build mixed-model and finding out most responsible factors that contribute more to the votes differences between similar questions or answers in terms of cloned code snippets, we need to collect all the factors first. We extract and collect most of the factors from the fast answer paper of S. Wang et al. [10] where the authors analyzed 46 factors along with 4 dimensions (e.g., question, asker, answer, answerer) for getting fast answers and build logistic regression model to understand the relationship between the studied factors and the needed time to get an accepted answer [10]. Therefore, we follow same approach of this paper for extracting the factors in terms of four dimensions. S. Wang et al. collected factors only for answers getting fast answer [10], but here we collect factors for both questions and answers separately because we want to find potential factors affecting similar posts (i.e., questions or

answers). Our collected factors for question and answers will be discussed in the subsequent subsection separately.

A. Potential factors affecting similar question's votes

In terms of questions, we collect 24 factors from questions that are related to text, tags, revision, votes and directly questions as well as asker itself. These factors will be listed bellow in different sections separately.

1) Questions related factors: In this section, we will discuss briefly 16 extracted questions related factor that are mainly considered from the question's perspective.

- **Questions_#View_Count:** It is a number of visited users on a specific post. When askers post questions, answerers view that post either for answering or find solution from the answers of that question for their problems.
- **Questions_Title_Length:** Length, this is the total length of the questions title in characters.
- **Questions_Creation_Date:** Date and time of any posted questions on the Stack Overflow.
- **Questions_Total_Length:** This factor indicates the total Length of whole question body including source code and all the HTML tags in the body in characters.
- **Questions_#Favourite_Count:** Favourite count indicates the popularity of the question on the Stack Overflow. So, the number of this factor show the popularity of a question.
- **Questions_Score:** This factor indicates that if the post quality is good and helpful users provide upvotes as positive score and downvotes for erotic or problematic post as negative score. Question score is the summation of positive and negative score.
- **Questions_#Answers_Contains:** Indicates number (count) of answers under each question.
- **Questions_#Tags:** Count the number of tags attached to each question.
- **Questions_CodeRatio_Percentage** This factor indicates the percentage of the code in questions from the whole question body.
- **Questions_#Link_Tags:** Number of hyperlink tags (i.e., "<a href =>") in the questions body.
- **Questions_#Italic_Tags:** Number of italic tags (i.e., ...) in the questions body.
- **Questions_Total_Code_Length:** Total code length in the question body in characters.
- **Questions_#Bold_Tags:** Number of italic tags (i.e., ...) in the questions body.
- **Questions_#Code_Snippets:** Number of code snippets (i.e., <code>....</code >tags) in the questions body.
- **Questions_#Comments:** Number of comments under each question in the comment or discussion section.
- **Questions_#Revision:** Number of revision or edition done by Stack Overflow users after the questions being posted.

2) Asker related factors: In this section, we will briefly discuss 8 extracted asker related factors that are mainly considered in terms of asker perspective.

- **Asker_Reputation_Score:** This factor, asker reputation score indicates the overall activities (i.e., asking, answering and editing the posts) on SO including positive and negative in terms of incentive system.
- **Asker_Total_UpVote:** Total number of up votes earned by the users throughout the period.
- **Asker_Total_DownVote:** Total number of up votes achieved by the users throughout the period.
- **Asker_#Posted_Questions:** Total number of posted question on SO by the users.
- **Asker_#Posted_Answers:** Total number of posted answers by the users.
- **Asker_Min/Median/Max/Mean/Average/25 Percentile_UpVote_EachPost:** Through this factor, we calculate the Min/ Median/ Max/ Mean/ Average/ 25 Percentile up votes of each posted questions or answers of the users on SO.
- **Asker_Min/Median/Max/Mean/Average/25 Percentile_DownVote_EachPost:** Through this factor, we calculate the Min/ Median/ Max/ Mean/ Average/ 25 Percentile down votes of each posted questions or answers of the users on SO.
- **Asker_#Revision_In_Questions:** Total number of revision or edit done by the asker in questions.
- **Asker_#Revision_In_Answers:** Total number of revision or edit done by the asker in answers.

B. Potential factors affecting similar answer's votes

In terms of answers, again we collect 21 factors from questions that are related to text, tags, revision, votes and directly answer as well as answerer itself. Actually, we would make list and discuss of those factors bellow in different sections separately. As all of the factors are similar with questions related factors, we will not describe again in this section. Therefore, we will just only mention the name of those factors that are similar or common factors for both questions and answers. So, there are 21 factors for answers which be categorized in two dimension such as answer related factor, answerer related factors like questions.

- **Answer related factors:** There are 11 factors which are related to the answers such as Answer_Creation_Date, Answer_Length, Answer_Score, Answer_#Bold_Tags, Answer_#Code_Snippets, Answer_#Comments, Answer_#Revision, Answer_Code_Ratio_In_Percentage, Answer_#Link_Tags, Answer_#Italic_Tags, Answer_Total_Code_Length
- **Answerer related factors:** There are 10 factors which are related to the answerers such as Answerer_#Revision_In_Questions, Answerer_#Revision_In_Answers, Answerer_#Reputation_Score, Answerer_#Total_Up_Votes, Answerer_#Total_Down_Votes, Answerer_#Posted_Questions, Answerer_#Posted_Answers, Answerer_Min/ Mean/ Max/ Variance/ Average/25 Percentile_UpVotes_EachPost, Answerer_Min/ Mean/ Max/ Variance/ Average/25 Percentile_DownVotes_EachPost.

C. Approaches for populating factors data

As we mentioned in Section-V, we follow three ways to populate our factors data. Now, in this section we will discuss these three approaches. First, for populating 7 factors data, we developed our third java tool (ToolThree [9]) that will extract questions related factors such as total code length, code ratio or percentage of code in questions body, total number of different types of tags (e.g., bold, italic, link) in each post, number of code snippets in each post (i.e., question and answer). Once we developed the ToolThree, we extracted these 7 factors in CSV file format for importing into the MSSQL and joining with main cloned data-sets to combine all the factors in the same table. Second, we also populate two-third (2/3) factors data by MSSQL server using simple, complex and advance level of SQL query (available here: [10]). In addition, we also use T-SQL for populating the factor's data. Finally, we use R Studio (scripts are available here inside in R projects: [11]) for calculating some other factors such as calculating Max, Min, Mean, Median, Variance, and 25 percentile votes of each post of the poster (i.e., asker or answerer). Here, we use R Studio for such kind of calculation because it is very simple to calculate these kind of terms by only using APIs of R packages.

VI. MODEL CONSTRUCTION FOR CLONED QUESTIONS AND ANSWERS

This section presents and discusses the approach along with result of mixed modeling in term of our two research questions. For each research question, we describe the approach followed to address the research question, and the results as well.

A. Mixed Model: Pre-processing of Populated Factors Data

At first, we did data wrangling or cleaning to our populated data for proceeding to the next steps. So, we replaced the NULL and empty values with zero (0), and removed white space is there any in the cloned data after populating. Afterward, we decided to build model in order to find out the more responsible factors that affect the vote differences among the cloned questions and answers. Therefore, we chose mixed-model because our clone groups are not very large. Actually, we manually check the cloned groups by pairwise using SQL query and we found that single question or answer only has several clone, but not too many clone. But Needless to say that, during clone detection processing NiCad generate only 4 huge clone groups with 29616, 5908, 3812 and 9372 clone code snippets for both in questions and answers, hence we manually removed those unexpected and big cloned groups from our data-sets to avoid biased results. Afterward, we decided to do mixed modeling with that data-sets.

In addition, mixed-model estimates the effects of one or more explanatory variables on a response variable. The output

of a mixed model will give us a list of explanatory values, estimates and confidence intervals of their effect sizes, p-values for each effect. We use a mixed model instead of a simple linear model because we have a variable or factor that describes our data sample as a subset of the data we have collected and populated.

B. Mixed Model: Technique Selection

Once we cleaned unwanted data, extracted factors and populated all the factors data, we decided to build the mixed model for both cloned questions and answers. Before doing mixed modeling, we pre-process the populated cloned data-sets for finding suitable technique that fit into our data-sets. Actually, we check the data distribution of each explanatory variables (i.e., factors/ metrics) for modeling by plotting histogram. Then we found that the data distribution of almost all explanatory variables (except one or two variables) are non-normal (i.e., left skewed or right skewed). Hence, we select glmmPQL (Generalized Linear Mixed Models via PQL) technique [12] to fit out mix model for our non-normal data [13]. Here, PQL mean Penalized Quasi Likelihood which is a flexible technique that actually deal with the abnormal or non-normal data, unbalanced design, and crossed-random effects [13]. Therefore, we finally move forward with PQL technique to find out the important factors through a list of independent (i.e., explanatory variable or factors), estimates and the confidence level of their effect size and p-value for each effect. For glmmPQL technique we need to load MASS package [13] in R studio.

C. Mixed Model: Cloned Questions and Answers

As we chose our glmmPQL technique, we build our mixed model to find out most important factors (i.e., explanatory variables) that have relationship with the votes (i.e., response variable) differences among the cloned questions and answers. Then we will answers our two research questions as follow:

RQ1: What are the factors that affects the vote differences of each cloned questions (i.e., cloned in terms of code snippets)?

Approach By this PQL techniques, we will find the correlation between the dependent and independents variables that affect the vote differences among cloned questions on the Stack Overflow. In this PQL technique, we consider votes of cloned questions as response variable (i.e., dependent variable) and all other factors are considered as the explanatory variables (i.e., independent). Additionally, we randomly change the fixed effect along with the family (e.g., gaussian) in the PQL technique and we use *library(nlme)* from R packages [14]. We adjust fixed and random effect several times. We also set the distribution to log-normal and set the family to gaussian (i.e., another word for normal) and the link to log as well. In

addition, before modeling we added 1 to both independent and dependent variables to eliminate zero values that avoids the error during the modeling.

This model calculate the p-value to determine the significance of the model. We focus on the p-value to know the statistically significance variables because p-value less than 0.05 (typically ≤ 0.05) indicates statistically significant. It also indicates strong evidence against the null hypothesis, as there is less than a 5% probability the null is correct. Therefore, we only focus on those variables who have p-value less than 0.05 (typically ≤ 0.05) and ignore those variable who have p-value higher than 0.05 (>0.05) in this model. We ignore those variables who have p-value higher than 0.05 because they does not have relation to the response variable and they does not have any impact on the vote differences of cloned questions. Besides, a p-value higher than 0.05 (>0.05) is not statistically significant and indicates weak evidence against the null hypothesis. This means we fail to reject the null hypothesis and cannot accept the alternative hypothesis. Finally, the factors which are not statistically significant (i.e., p-value higher than 0.05) in the modeling, we just ignore those variable after applying Anova() function from the model to avoid the biased results and then again we build the model with statistically significant variables that improve our results. We frequently use summary() and Anova() functions for checking our results in the model only with the statistically significant variables.

Resutls When apply Anova() function to the results of our model, Anova() functions does Wald χ^2 test, which tell us how confident we are of our estimate of the effect of explanatory variables. The larger the Wald χ^2 value, the larger the impact of a factor on the vote differences. Table IV shows the factors list that have the impact on the vote differences of the similar questions on the Stack Overflow. As a result, we can say that the explanatory (i.e., independent) variable marked with stars (*) have effect on the cloned questions and create vote differences among the cloned questions in each group. In terms of results of our model, we notice that there are 12 factors have impact on the vote differences of the cloned questions on the Stack Overflow.

Overall, we found 12 most accountable factors of cloned questions (i.e., cloned in terms of code snippets) through our mixed modeling that affect their (i.e., similar answers) vote differences. Therefore, these 12 factors are considered during voting to the similar questions by the Stack Overflow users that makes vote differences among the cloned answers. Table-IV shows all the factors list that are considered by the users while voting and these factors are responsible for vote differences on the Stack Overflow.

RQ2: What are the factors that affects the vote differences of each cloned answers (i.e., cloned in terms of code snippets)?

Approach Basically, we follow the same approach of RQ1.

---

[12]https://rdrr.io/cran/MASS/man/glmmPQL.html

[13]https://cran.r-project.org/web/packages/MASS/index.html

[14]https://cran.r-project.org/web/packages/nlme/index.html

Factor	Chisq	Df	Pr(>Chisq)
Questions_#View_Count	2421.3314	1	<2.2e-16 ***
Questions_#Favourite_Count	9398.0137	1	<2.2e-16 ***
Questions_#Code_Snippets	243.8100	1	<2.2e-16 ***
Questions_#Revision	506.2181	1	<2.2e-16 ***
Questions_#Link_Tags	14.6679	1	**0.0001282 *****
Questions_#Tags	9.9967	1	0.0015682 **
Questions_CodeRatio_Percentage	336.0658	1	<2.2e-16 ***
Questions_#Answers_Contains	508.8827	1	<2.2e-16 ***
Asker__Total_DownVotes	77.7744	1	<2.2e-16 ***
Asker_#Posted_Questions	72.9072	1	<2.2e-16 ***
Asker_#Posted_Answers	73.3646	1	<2.2e-16 ***
Asker_Reputation_Score	161.9860	1	<2.2e-16 ***

TABLE V
CLONED ANSWER'S FACTORS: AN OVERVIEW OF THE RESULTS OF OUR
MIXED MODELING USING glmmPQL TECHNIQUE

Factor	Chisq	Df	Pr(>Chisq)
Answer_length	16.183	1	**5.752e-05 *****
Answer_#Code_Snippets	178.222	1	<2.2e-16 ***
Answer_#Comments	21.129	1	**4.293e-06 *****
Answer_#Revision	999.592	1	<2.2e-16 ***
Answer_#LinkTags	113.283	1	<2.2e-16 ***
Answer_Total_Code_Length	16.726	1	**4.318e-05 *****
Answerer_Reputation_Score	61.282	1	**4.946e-15 *****
Answerer_Total_Down_Votes	1043.758	1	<2.2e-16 ***
Answerer_Posted_Questions	1916.234	1	<2.2e-16 ***
Answerer_Posted_Answers	1827.784	1	<2.2e-16 ***
Answerer_#Revision_In_Answers	131.795	1	<2.2e-16 ***
Answerer_Min/Mean/Max/Variance/ Average/25 Percentile _UpVotes_EachPost	4647.572	1	<2.2e-16 ***
Answerer_Variance_Down_Votes	129.151	1	<2.2e-16 ***

In RQ2, we also apply this glmmPQL technique individually. Therefore, we will not discuss the same approach here in this RQ2. We will only discuss the result from the cloned answers data-sets after modeling by using the glmmPQL technique.

Resutls We also apply Anova() function to the results of our model for answers and did same things like cloned questions in RQ1, hence we will not describe everything here in this RQ2. Table V shows the factors list that have the impact on the vote differences of the similar questions on the Stack Overflow. As a result, we can say that the explanatory (i.e., independent) variable marked with stars (*) have effect on the cloned questions and create vote differences among the cloned answers in each group. In terms of results of our model, we notice that there are 13 factors have impact on the vote differences of the cloned answers on the Stack Overflow. Needless to say that for each post of the answerers, we calculated min, median, max, variance, 25 percentile of up votes and consider these each terms as explanatory variable in the modeling and we also noticed that all of them are statistically significant. But in the Table-V, we show the result as a single factors because their Pr(>Chisq) value are same and all are statistically significant.

Overall, we found 13 responsible factors of cloned answers (i.e., cloned in terms of code snippets) through our mixed modeling that affect their (i.e., similar answers) vote differences. That means these 13 factors are considered during voting to the similar answers by the Stack Overflow users that makes vote differences among the cloned answers. Table-V shows all the factors list that are considered by the users while voting and these factors are liable for vote differences on the Stack Overflow.

VII. DISCUSSION

This quantitative study shows that online code cloning occur on Stack Overflow for both questions and answers. According to our experience and the case study result, developers re-use code snippets whenever needed either for posting questions and answers on the Stack Overflow, technical Q&A website. From our motivation, developers

are copying and pasting code snippets for different purposes and generating duplicate questions and answers on Stack Overflow in terms of clone code snippets. So, the scope of this paper is that although the cloned questions and answers are similar though their code snippets, why their vote are different to each other. In other word, although programmers are posting duplicate posts (i.e., questions and answers) on Stack Overflow at different time period, we observe that those similar questions and answers have vote variance. Therefore, we investigate all the responsible factors by building mixed modeling (i.e., PQL technique of mixed modeling) and confirmed 25 factors that are responsible for their vote variances. As a result, after investigating the issue of vote differences, we found 12 factors for questions and 13 factors for the answers that are affecting their vote. This discussion draw the conclusion since we extracted cloned questions and answers from Stack Overflow and detected all the responsible factors that affect the vote differences through mixed modeling.

Although this study contains some threats to validity that we will discuss in Section VII-A. In the future, we plan to study code reuse issues in other Q&A websites and in large-scale open-source systems.

A. Threats to Validity

Internal Validity Threats From the beginning to the end, we applied different kinds of tools and technologies, even we personally developed three java tools (i.e., ToolOne, ToolTwo, and ToolThree) throughout the research. First, we developed java tool (ToolOne) for extracting code snippets of "java" tags from Stack Overflow. However, this tool could not identify specific java related code snippets when askers provide other languages source code such as C, C++, XML, SQL and so on as an example or supportive code in the same post. Hence, the collection of code snippets were not only from java but also from other languages, although asker used "java" tags on the posts. As a result, the code

snippets we collected for clone detection leads to more error rate by the NiCad clone detection tool. Second, during the clone detection, NiCad could generate wrong clone with 70% similarity by using Type-3 settings of NiCad, therefore we checked the clone manually and we found several false positive clone. In addition, as we set 10 LOC (Lines of Code) as a threshold for cloning size in the NiCad, the accurateness of the cloned code would be biased because AST parser of NiCad compare the sub-tree of the java source code. Therefore, wrong cloned may be generated, although there are not exact or 100% matched between the cloned codes. Finally, during the investigation of responsible factors of the vote differences, the results of the mixed-model could be more accurate if we only use the clone specific data-sets for each groups. But in this study, we use all the raw data-sets instead of summarizing the clone group specific data-sets for both questions and answers.

External Validity Threats We chose Stack Overflow for our case study because it is very popular online open source technical Q&A website all over the world with 16.5 million questions, 26 million answers, 9 million accepted answer, and 10 million users by the end of 2018 and data dumps are also available on their official website[15]. This website contain wide range of programming languages with code snippets as an example and java is very popular programming language throughout the word. Therefore, in terms of code snippets, we downloaded a full data dump and extracted Java questions that have accepted answers and answers that are accepted and they are the most likely ones to be reused by the developers. Our result and finding may not be generalized to all programming languages and all posts on Stack Overflow.

VIII. RELATED WORK

In this section VIII, we discuss related works that investigated Q&A websites for finding clone code snippets. We already know that SO is very popular online technical Q&A website throughout the world. S. A Licorish et at. [14] investigate code reuse between Stack Overflow to Open Source Software (OSS). When developers reusing code from Stack Overflow for their OSS projects may violate licensing agreements if their attributions are not correct. The authors analyzed CCFinder and apply this tool for finding the cloned code snippets. D. Yang et al. [15] wished to study how developers use Stack Overflow code snippets in their projects and is it possible to find out the copied Stack Overflow code snippets in the developers real projects (e.g., GitHub projects). C. Ragkhitwetsagul et al. [16] studied online code clones that are copied from software projects or online sources to Stack Overflow as examples. After the code has been copied from Stack Overflow to personal software projects, they can be harmful code snippets (i.e., they suffer from being outdated or violating the original software license) because a lack of checking mechanism. M.A Nishi et al.

---

[15]https://archive.org/details/stackexchange

[1] investigate similar or clone code snippets among two popular online programming resources such as the Stack Overflow Q&A site and software development free online programming tutorials websites. The authors were concern to understand the frequency of reusing the source code between these two sources, to acquire knowledge about why developers copy code snippets from one online source to the other, and to understand the evolution of duplicated source code over time. Le An et al. [5] investigate whether developers respect license terms (i.e., *Creative Commons Attribute-ShareAlike 3.0 Unported*) when reusing source code from Stack Overflow posts (and the other way around) and find the license violation by the developers while copy and paste code. R. Abdalkareem et al. [1] understand the impact of code reuse from Stack Overflow by an exploratory study which focusing on code reuse from Stack Overflow in terms of mobile apps and further study to know how much, why, when, and who reuses code. Moreover, to understand the potential implications of code reuse, we examine the percentage of bugs in files that reuse Stack Overflow code.

IX. CONCLUSION

The question and answer (Q&A) website, Stack Overflow, is very popular and overly used online website where programmers discuss programming problems, share expertise, and exchange ideas as well as provides a code snippets in the posts to support their questions and answers as an example. It also allow users to reuse the code snippets within the Stack Overflow community for both questions and answers. Therefore, we conduct a case study with only Java questions (i.e., that have accepted answers) and accepted answers separately on Stack Overflow. During the clone detection process with NiCad, we found 60,079 (60K) java unique questions that comes from 18,701 clone groups (i.e., 2,185 clone groups from blocks and 16,516 clone groups from functions). That means this 60,079 (60K) java questions are similar to each others in terms of cloned code snippets. We accounted that 20% of the java questions are cloned from 291,427 (0.29 M) java questions that have at least 10 lines of code (LOC). Similarly, in terms of accepted answers clone we detected 16,316 (16.3K) java unique accepted answers that comes from 3,656 clone groups or classes (i.e., 852 clone groups from blocks and 2,804 clone groups from functions). This 16,316 java accepted answers are cloned to each other with regards to the clone code snippets. We accounted that 12.24% of the java accepted answers are cloned from 133,243 (0.13 M) java accepted answers that have at least 10 lines of code (LOC). Needless to say, the error rate of NiCad clone detector was really high (68% for questions and 77% for answers), otherwise we could get more clones from the extracted java code snippets. Hence, to draw conclusion, developers are copy pasting the source codes on the Stack Overflow for asking questions and providing answers along with example code snippets.

On the other hand, we noticed the vote differences among the cloned questions and answers, hence we wish to study the factors which affects the vote differences among the cloned

questions and answers. Therefore, we planed to do mixed modeling for finding out the most responsible factors of vote differences. Finally, we found total 25 factors (i.e., 12 factors for questions and 13 factors for answers).

The findings of our paper indicates that when developers copy paste code snippets they do not concerns about the cloning but when the users of Stack Overflow provides votes to the similar questions and answers they consider 25 factors before voting that makes vote differences among the cloned questions or answers.

REFERENCES

[1] M. A. Nishi, A. Ciborowska, and K. Damevski, "Characterizing duplicate code snippets between stack overflow and tutorials," in IEEE International Working Conference on Mining Software Repositories, Montreal, QC, Canada, 2019, vol. 2019-May, pp. 240–244.

[2] R. Abdalkareem, E. Shihab, and J. Rilling, "On code reuse from stackoverflow: An exploratory study on android apps," Information and Software Technology, vol. 88, pp. 148–158, 2017.

[3] D. Yang, P. Martins, V. Saini, and C. V. Lopes, "Stack Overflow in Github: Any Snippets There?," CoRR, vol. abs/1705.01198, 2017.

[4] https://www.quantcast.com/stackoverflow.com [Accessed January, 2017]

[5] L. An, O. Mlouki, F. Khomh, and G. Antoniol, "Stack Overflow: A Code Laundering Platform?," CoRR, vol. abs/1703.03897, 2017.

[6] J. Svajlenko and C. K. Roy, "Evaluating modern clone detection tools." in Proceedings of the 30th International Conference on Software Maintenance and Evolution (ICSME 2014). IEEE, 2014, pp. 321–330.

[7] J. R. Cordy and C. K. Roy, "The NiCad clone detector," in Proceedings of the 19th International Conference on Program Comprehension (ICPC 2011). IEEE, 2011, pp. 219–220.

[8] S.M. Nasehi, J. Sillito, F. Maurer, C. Burns, What makes a good code exam- ple?: A study of programming Q&A in StackOverflow, in: Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM), 2012, pp. 25–34.

[9] C.K Roy, J.R. Cordy and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach", Sci. Comput. Program. 74(7), pp. 470-495, May 2009.

[10] S. Wang, T.-H. Chen, and A. E. Hassan, "Understanding the Factors for Fast Answers in Technical Q&A Websites: An Empirical Study of Four Stack Exchange Websites," in Proceedings of the 40th International Conference on Software Engineering, New York, NY, USA, 2018, pp. 884–884.

[11] J. R. Cordy and C. K. Roy, "The NiCad clone detector," in Proceedings of the 19th International Conference on Program Comprehension (ICPC 2011). IEEE, 2011, pp. 219–220.

[12] Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J.-S. S. (2009). Generalized linear mixed models: a practical guide for ecology and evolution. Trends in ecology & evolution, 24(3), 127135.

[13] A Practical Guide to Mixed Modeling in R [Online]. Available: https://ase.tufts.edu/gsc/gradresources/guidetomixedmodelsinr/mixed%20model%20guide.html

[14] A. Lotter, S. A. Licorish, B. T. R. Savarimuthu and S. Meldrum, "Code Reuse in Stack Overflow and Popular Open Source Java Projects," 2018 25th Australasian Software Engineering Conference (ASWEC), Adelaide, SA, 2018, pp. 141-150. doi: 10.1109/ASWEC.2018.00027

[15] D. Yang, P. Martins, V. Saini, and C. V. Lopes, "Stack Overflow in Github: Any Snippets There?," CoRR, vol. abs/1705.01198, 2017.

[16] C. Ragkhitwetsagul, J. Krinke, M. Paixão, G. Bianco, and R. Oliveto, "Toxic Code Snippets on Stack Overflow," CoRR, vol. abs/1806.07659, 2018.