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

***Abstract*—Code clones 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 copying and pasting the code snippets for asking questions and answering the questions within the Stack Overflow. In this paper, we conduct a case study with only Java questions (i.e., that have accepted answers) and accepted answers separately on Stack Overflow, to investigate i) similar questions and answers in terms of cloned code snippets ii) vote differences among the cloned questions or answers iii) factors that could have been affected vote differences between cloned questions or answers iv) which factors exactly contribute more to the vote differences within duplicate questions or answers by building mixed-modeling. In terms of cloned questions, we found 20% that means 60,079 (60K) java questions are cloned to each other from 291,427 (0.29 M) java questions that contains 0.37M code snippets. In terms of cloned answers, we found 12.24% that means 16,316 (16.3K) java accepted answers are cloned to each other from 133,243 (0.13 M) java accepted answers that contains 0.13M code snippets. Furthermore, we collect around 45 factors (i.e., metrics) in total for both questions and answers that could affect vote differences and build mixed-model to finding out which factors contribute more to the vote differences among all the factors to the cloned questions or answers. In addition, there are several factors (TODO : PROVIDE THE NUMBER) that affect vote differences of those cloned questions or answers. So, this study reveals that developers commonly use Stack Overflow to ask clarifying questions about code they reused from Stack Overflow, and again copy code snippets from Stack Overflow to provide answers to questions, but different factors are responsible for vote differences of similar questions or answers .**

***Index Terms*—component, formatting, style, styling, insert**

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 most popular technical Q&A website, Stack Overflow. By this way, SO accumulated around 16.5 Millions questions, 26 Millions answers, 9 Millions accepted answer, and 10 Millions 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 contains 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 [6] or copied from Stack Overflow to android apps for development.

Although, the code copying and pasting trend have been running in a continuous 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 snippets for multiple times for multiple questions. Therefore, developers also post answers by searching online tutorial or from their own 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 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

I, thanks to professor Dr Tse-Hsun (Peter) Chen for helping me throughout the research projects .

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 1709389 (1.7 M) Java question that have accepted answers. From 0.91M java questions, we extracted 1315097 (1.3 M) code snippets for clone detection and further analysis. In terms of answers, we analyze 517482 (0.51 M) java accepted answer that have have at least 10 LOC (i.e., lines of code) after filtering from total 1707497 (1.7 M) Java accepted answers. From 0.51M java accepted answers, we extracted 656180 (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)?**
  In terms of questions, from 1315097 (1.3 M) code snippets of java questions that 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.
- **RQ2: What are the factors that affects the votes differences of each cloned Answers (i.e., cloned in terms of code snippets)?**
  In terms of accepted answers, from 656180 (0.65 M) code snippets of java accepted answers, we got 852 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.

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 describe each step in our data processing approach below. The corresponding data and scripts are available online at:https://github.com/MMH08/SOEN-7481-Course-Project.

A. Data Collection : code snippets corpus

In order to set up our environment for study, we obtained the StackOverflow data dump (published dec 2018) in XML format and download the data dump from SO official website in XML format. Then we imported the XML data into MSSQL for further analysis or mining the SO data. The data dump contained 42736511 posts, including 16635796 questions and 26003926 answers until 2018.To perform our case study, we extracting Java questions and answers from StackOverflow . 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. [**TODO add a questions thread with label**]

B. Data Filtering Approach of Questions and Answers

For the next mining step, we extract the relevant posts for our study context using simple or advanced SQL query to MSSQL. 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 (3229726 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 StackOverflow, we further filtered our dataset 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 StackOverflow 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 least 10 LOC (lines of code) after developing java code snippets extractor tool [link]. Therefore, finally we got 910936 java posts that have accepted answers, example code snippets and have 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

TABLE I
QUESTIONS: JAVA POSTS FILLETING APPROACH OF 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:JAVA POSTS FILLETING APPROACH OF 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)	-

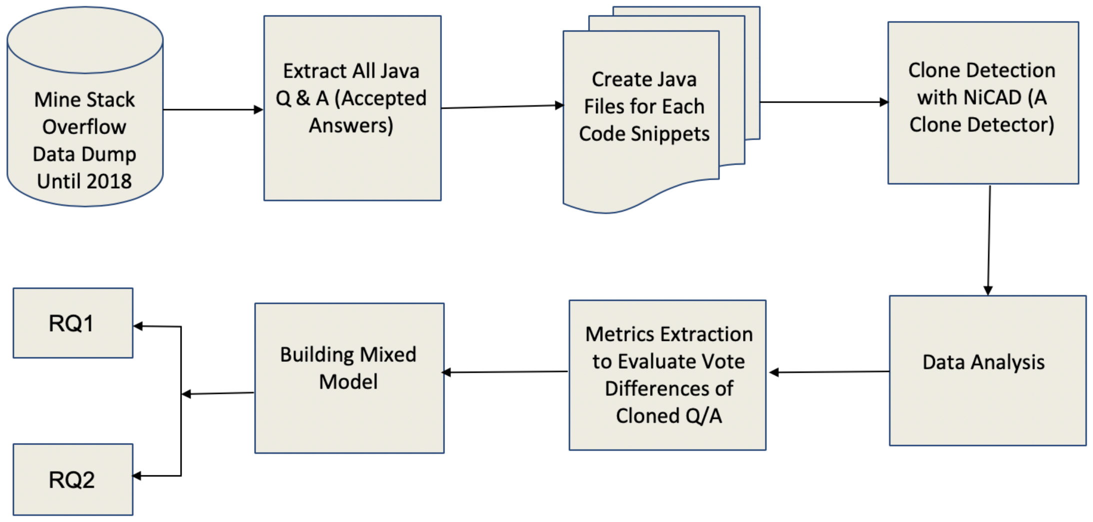
less than 10 lines of code users may share a simple one line of variable, if or for loop statement or a few lines of block of code that may flag common for all the example code snippets. Therefore, selecting 10 lines of code restricted the size of our datasets and it also decrease the false positive (FP) rate by the clone detection tool to detect as a clone. Hence, 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 are accepted of java questions. 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 1 summarizes how we filter questions and get the final number of posts after finishing the filtering. In terms of accepted answers, Table 2 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 tool [LINK] which extract the each code snippets into each java files (i.e., .java). Therefore, after finishing the identification and exporting phases of all Stack Overflow posts that contained example code blocks, we followed a straightforward and traditional techniques to identify the source code from each Stack Overflow posts. Actually, the whole body of a questions or answers are comprised of the HTML code. 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 a java tools [TODO: link of tools] 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 really impossible to remove all those irrelevant code snippets from the same post (i.e., questions or answers) that are not related to java code 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 develop the Java tool [link] for extracting code snippets. But those errors were easy to solve by filtering and replacing information using several if else conditions in the tool and we were successful to solve some of them errotic code using our code extractor java tool 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 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 example SO post associated with postid 36731552 [TODO : add link]:

- **Message or Comment in Code Block:** SO users sometimes provide message in the code without using double slash "//" which lead to parsing error by the clone detector. This post is an ideal example of these type



Mine Stack Overflow Data Dump Until 2018
Extract All Java Q & A (Accepted Answers)
Create Java Files for Each Code Snippets
Clone Detection with NiCAD (A Clone Detector)
Data Analysis
Metrics Extraction to Evaluate Vote Differences of Cloned Q/A
Building Mixed Model
RQ1
RQ2

Fig. 1. Overview of our data processing approach

of problematic code and this post (i.e., questions) is associated with post Id : 36731552

- **Semicolon Missing:** 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 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 at the time of asking and answering questions.
- **Others:** Call functions without developing functions definitions in the code and use "..." in the middle of the code and many more because we found a lot of types of errotic code format within the code snippets.

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

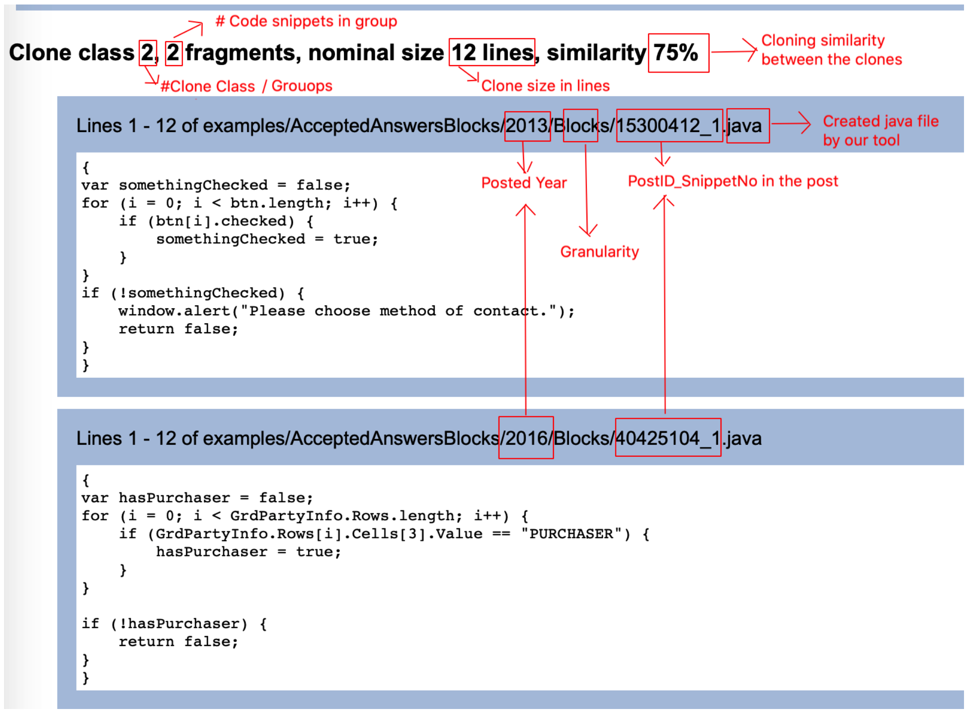
However, after following this ideal approach, the error rate of the clone detection using NiCad is still high which accounted 68% for questions and 77% for the answers from 1.3 M and 0.65 M code snippets respectively. This is because of the problematic code on Stack Overflow posted by users and other 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).

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)



# Code snippets in group
Clone class 2, 2 fragments, nominal size 12 lines, similarity 75%
Cloning similarity between the clones
#Clone Class / Grouops
Clone size in lines
Created java file by our tool
Posted Year
PostID_SnippetNo in the post
Granularity

Lines 1 - 12 of examples/AcceptedAnswersBlocks/2013/Blocks/15300412_1.java

```
{
var somethingChecked = false;
for (i = 0; i < btn.length; i++) {
    if (btn[i].checked) {
        somethingChecked = true;
    }
}
if (!somethingChecked) {
    window.alert("Please choose method of contact.");
    return false;
}
}
```

Lines 1 - 12 of examples/AcceptedAnswersBlocks/2016/Blocks/40425104_1.java

```
{
var hasPurchaser = false;
for (i = 0; i < GrdPartyInfo.Rows.length; i++) {
    if (GrdPartyInfo.Rows[i].Cells[3].Value == "PURCHASER") {
        hasPurchaser = true;
    }
}

if (!hasPurchaser) {
    return false;
}
}
```

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

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 we passed the code snippets through the NiCad clone detection tool, it generates the result in terms of block and functions in HTML files 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) 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 [TODO: tool link] 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, the HTML parser tool 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 generated by NiCad. We import those cloned datasets 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. Because we considered two posts are duplicate or similar either questions or answers in terms of cloned code snippets. Therefore, we extract post (i.e., questions or answers) information of cloned code snippets from HTML formatted results and find more factors (i.e., 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 data in CSV file. 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 online [ADD GITHUB LINK HERE] as open source and convenience of future researchers. 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 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 vote differences among the group members. But the most 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. 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 case study would create biased results, we removed those data from our study to achieve the better result from our case study and model building as well.

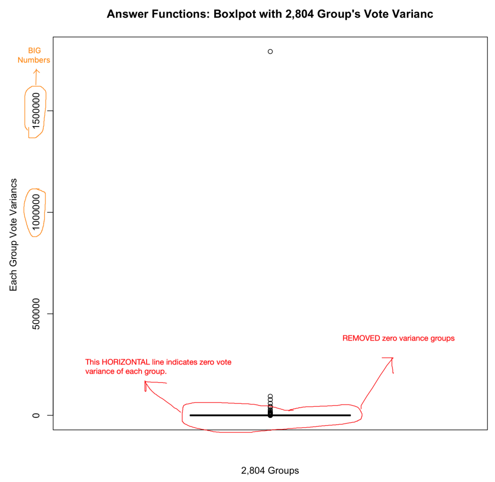


Fig. 3. Boxplots with BIG number that creates problem visualizing properly for smaller numbers in the figures to understand the data distributions.

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

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

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 easily. Finally, we draw boxplot with the normalized variance and SNE normalize the vote variances within the limit 0-1 has shown in the Y-axis in the figure. Figure-5 represents the better visualization of vote variance compared to 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 factors data, 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 factors mostly considering the fast answer paper of S. Wang et al. [10] where the authors analyze 46 factors along with 4

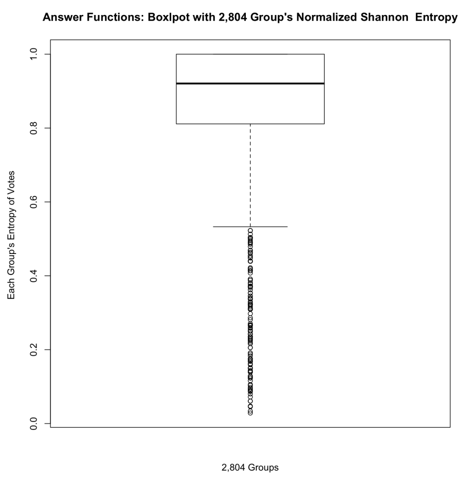


Fig. 5. In order to know the vote differences among the group member of the cloned questions or answers, we applied Shannon Normalized Entropy (SNE). SNE helps to better visualization of vote differences after applying to each cloned groups.

dimensions (e.g., question, asker, answer, answerer) for getting fast answers and build logistic regression model in order 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_#ViewCount:** 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 =></a >") in the questions body.
- **Questions_#Italic_Tags:** Number of italic tags (i.e., <em>....</em>) 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., <strong>...</strong>) 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 discuss briefly 8 extracted asker related factors that are mainly considered from 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 [PROVIDE GITHUB LINK HERE] 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), number of code snippets for all posts (i.e., questions and answers). Once we developed the tool, we extracted these factors in CSV file format for importing into the MSSQL and join with main datasets 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 [ADD QUERY LINK]. In addition, we also use T-SQL for populating the factor's data. Finally, we use R Studio for calculating some other factors such as calculating Max/Min/Mean/Median/variance/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. Building Mixed Model

In this section, we will discuss the approach of building mixed model in order to find out the more responsible factors that affect the vote differences among the cloned questions and answers in terms of code snippets. 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-set to avoid biased results. Afterward, we decided to do mixed modeling with this data-set.

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.

Basically, we build two mixed model, one for questions and another for answers separately. As we collected all the metrics or factors for building mixed-model which discussed in Section-V, now here in this section, we will build mixed modeling to find most important factors (i.e., explanatory variables) that have correlation with the votes (i.e., response variable) differences among the cloned questions and answers.

A. Mixed Model Building for Questions

Once we cleaned unwanted data, extracted factors and populate all the factors data for questions, we decided to build the mixed model for cloned questions in terms of code snippets. Before doing mixed modeling, we pre-process the populated cloned questions data.

VII. Mixed Model Result

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks …". Instead, try "R. B. G. thanks…". Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first …"

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

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] Stack Overflow in Github: Any Snippets There?.

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

[5] Stack Overflow: A Code Laundering Platform?

[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] Shaowei Wang, fast answers paper with peter chan

[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] 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.