SENTIMENT ANALYSIS ON JAVA SOURCE CODE IN LARGE SOFTWARE REPOSITORIES

by

Vinayak Sinha

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Computing and Information Systems

YOUNGSTOWN STATE UNIVERSITY

May, 2016

SENTIMENT ANALYSIS ON JAVA SOURCE CODE IN LARGE SOFTWARE REPOSITORIES

Vinayak Sinha

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Vinayak Sinha, Student Date

Approvals:

Bonita Sharif, Thesis Advisor Date

Alina Lazar, Committee Member Date

John Sullins, Committee Member Date

Sal Sanders, Associate Dean of Graduate Studies Date

Abstract

While developers are writing code to accomplish the task assigned to them, their sentiments play a vital role and have a massive impact on quality and productivity. Sentiments can have either a positive or a negative impact on the tasks being performed by developers. This thesis presents an analysis of developer commit logs for GitHub projects. In particular, developer sentiment in commits is analyzed across 28,466 projects within a seven-year time frame. We use the Boa infrastructure's online query system to generate commit logs as well as files that were changed during the commit. Two existing sentiment analysis frameworks (SentiStrength and NLTK) are used for sentiment extraction. We analyze the commits in three categories: large, medium, and small based on the number of commits using sentiment analysis tools. In addition, we also group the data based on the day of week the commit was made and map the sentiment to the file change history to determine if there was any correlation. Although a majority of the sentiment was neutral, the negative sentiment was about 10% more than the positive sentiment overall. Tuesdays seem to have the most negative sentiment overall. In addition, we do find a strong correlation between the number of files changed and the sentiment expressed by the commits the files were part of. It was also observed that SentiStrength and NLTK show consistent results and similar trends. Future work and implications of these results are discussed.

Acknowledgements

First of all, I would like to express my gratitude to my advisor, Dr. Bonita Sharif, whose expertise, understanding, and patience, added significantly to my graduate experience. I appreciate her vast knowledge and skills in many areas and her guidance, which has been a key motivational factor towards completion of my research. Further, I would like to thank the esteemed members of my committee, Dr. Alina Lazar and Dr. John Sullins for the assistance they provided at all levels of the research project.

A very special thanks goes out to my family without whose motivation and encouragement I would not have considered a graduate career. They are among the many people who truly made a difference in my life. I thank all the members who directly or indirectly helped me to complete my research work.

In conclusion, I recognize that this research would not have been possible without the financial assistance from the Department of Computer Science and the STEM College at YSU. I express my gratitude to them for supporting me during my graduate studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

There is an increasing amount of research in the software engineering community dealing with sentiment and the emotional aspect of software development. Sentiment analysis or opinion mining was initially developed as an automated method of extracting sentiment polarity from short texts posted online such as movie reviews, product reviews, microblogs and tweets. Recently, this method was adopted by the software engineering community and applied to different software engineering artifacts such as commit logs [1], question and answer posts and online mailing list messages [2]. The sentiments a developer projects during development are important as they could have an impact on productivity.

The work presented in this thesis resembles the work by Guzman et al. [1] with some important differences. First, we analyze developer sentiment in commit logs on a much larger set – 2,251,585 commit logs. Second, we also take a look at the number of files changed and map them to the sentiment expressed in the commits that the files were part of. We do this across the entire project's lifetime up until 2015.

1.1 Motivation

Guzman et al. analyzed 60,425 commit messages [1] from 90 top rated GitHub projects. They found Java projects tend to have more negative comments with distributed teams having more positive comments. They also found that Mondays had the most negative emotion associated with them. Murgia et al. conduct an exploratory study to have humans rate or agree on emotions in issue reports [3]. They found that developers do indeed express emotions and positive emotions had higher agreements between human raters. The goal was to eventually automate emotion mining in software artifacts. Jongeling et al. provide a good comparison of four sentiment analysis tools for software engineering research and also conduct an analysis of whether the tool sentiment matches a human evaluator's sentiment [4]. They found that the tools gave contradictory results when run on issue tracker data. They call for more tools targeting software engineering artifacts. The main motivation of this thesis is to be able to replicate these findings on a much larger dataset. In addition, we focus mainly on Java projects.

1.2 Contributions

The main contribution of this thesis is an empirical study to understand developers' sentiment emotions. The more we understand about developer emotions, the better we can support them by providing better tools for them during development. In addition, this research will also help management to understand the sentiments of their software developers and provide them a better work environment. This will result in increased productivity of the developers. We wrote Boa scripts [26][27] that we ran through the web-based Boa interface. This allowed us to download all the commit logs for Java projects from the GitHub Medium (September 2015) dataset available on Boa website. In total, we extracted 2,251,585 commit logs across 28,466 projects.

1.3 Research Questions

We seek to answer the following research questions:

- RQ1: What is the general developer sentiment in commit messages for GitHub projects?
- RQ2: What is the relationship between developer sentiment in commit messages and the day of the week the commit was made?
- RQ3: Is there a correlation between the number of changed files and developer sentiment?
- RQ4: How do existing popular sentiment analysis tools compare when applied to commit messages?

The first research question, RQ1 aims to understand the overall developer sentiment in commit logs on 2,251,585 commit logs. We split the data into three categories – projects having maximum number of commits, projects having average number of commits and projects having low number of commits. Top five projects from each category were selected for RQ2 which calculates the day of the week based on the commit date. The sentiments are distributed over the week to analyze and understand the relationship between developer sentiment in commit messages and the day of the week the commit was made. With every commit made on Github there are file(s) that gets added, deleted or modified; we also aimed to understand if there is any correlation between developer sentiment and the files that gets added, deleted or modified. We used SentiStrength for RQ1, RQ2 and RQ3; the main objective of RQ4 was to compare the output reported in RQ1, RQ2 and RQ3 using SentiStrength with the output obtained by

NLTK a Python library that determines the sentiment of a given text. We chose these two particular tools due to their popularity and good accuracy when used in prior research.

1.4 Organization

The thesis is organized as follows. The next chapter gives a brief introduction sentiment or opinion analysis and related work. Chapter 3 describes the dataset used and sentiment analysis tools that are used on the commit logs. The experiments conducted are explained in detail. Chapter 4 describes the results to our research questions. Finally, we discuss our results, and state our conclusions and future work in Chapter 5. Parts of the thesis have been published in our mining software repositories challenge paper entitled "Analyzing Developer Sentiment in Commit Logs", published at the 13th International Conference on Mining Software Repositories (MSR 2016) [5].

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter presents an overview of existing work in sentiment or opinion analysis. We first start with giving some terminology on sentiments and emotions; further we will discuss sentiment analysis performed in different areas.

2.1 Terminology

We describe some terminology that is used throughout this thesis. We use the dictionary meaning [6] for each of these terms.

- *Emotion*: An affective state of consciousness in which joy, sorrow, fear, hate, or the like, is experienced, as distinguished from cognitive and volitional states of consciousness.
- *Sentiment*: A mental feeling.
- *Affect*: To impress the mind or move the feelings of.

The above mentioned terms are related to each commit log which enables us to understand the mental feeling of the developer (sentiment), their emotional state and how it affects the productivity and accuracy of the task being performed by developers.

2.2 Sentiment in Tweets

Twitter has emerged as one of the most typical example where scholars, advertisers and political activists put forth their views and opinions in the form of status messages which are also termed as "*tweets*". Twitter also provides corporate industries and organizations to get customer reviews from across the globe in order to perform further analysis on the products or services provided by the organizations [7][8]. There are instances where real time Twitter sentiment analysis was performed; an example was 2012 U.S. Presidential Election. Hao Wang et al. [9] collected public reviews from Twitter about the U.S. presidential candidates for 2012 elections and used as inputs which were passed through a sentiment model which was designed to determine tweets sentiments (positive, negative, neutral, sarcastic, humorous or unsure). The motivation behind this study was to understand how these electoral events have affect on public opinions.

Based upon these scenarios and with the availability of advance technology in social media analytics a number of researchers and scholars are trying to understand the sentiments linked with the tweets being posted on Twitter. Over time, several tools have been developed which determine sentiments for each tweet that has been made, depending upon the number of inputs passed through the tools. The tools provide information for each input, which determines if the tweet is having positive, negative or neutral sentiments. The information gathered are further accumulated to determine if the sentiments were strong positive or strong negative [10]. Pfitzner et al. [11] analysis on the Twitter data suggests that information shared ("*tweeted*") on Twitter tend to be shared again ("*retweeted*") on Twitter, if they have high emotional or sentimental affect (very positive or very negative words determining sentiments) which was termed as *emotional divergence*. Further analysis also suggested that there was correlation between long texts and emotional divergence as long text tend to have very positive and very negative words. Kouloumpis et al. [12] collected set of three types of datasets namely Hashtagged data set, Emoticon data set and iSieve data set to understand sentiments on twitter messages. The motivation behind the study was to create training data with labels determined by hashtags and emoticon which are useful understand how are sentiments can be classified on messages posted on Twitter. Analysis was performed on the Hashtagged and Emoticon dataset; the initial process involved some data cleansing on the Hastagged dataset, which included removal of duplicate tweets and non-English tweets. The Emoticon dataset used was created by Go, Bhayani, and Huang at Stanford University. The tweets were categorized as either positive if they had emoticon ':)'or negative if they had emoticon ':('.

2.3 Sentiment in Mailing Lists

To share and exchange information over the internet, the most common method used is sending an email. The amount of emails sent across from our email address might be enough to understand one's thoughts and views. These emails keep a track of all your records creating detailed life-log which contain good and bad memories [13]. A number of researchers have tried to understand the sentiments that pass over with the email sent. Informal letters such as love letters, hate letters or suicide notes [14] have been analyzed to understand the mental state and sentiment of the person writing the email and hence providing a methodology to track emotions. Mohammad et al. [15] used the Enron email corpus data to analyze over 200,000 emails [16]; the data gathered also included meta data to determine timestamp and the email address of the emails sent or received. The motivation behind the research was to understand if the use of emotional words varies with the gender difference in corporate organizations [17][18]. Therefore, the genders of the author of the emails were identified to perform the analysis; also, there were instances where the gender could not be identified by the name and hence was categorized as "*unsure*". They observed after the analysis that emails sent by women and received by men contain more trust words and emails sent by men and received by women contain more joy words.

Emotional comparison based on gender has been analyzed at work places and in personal emails to understand how men and women are likely to react and maintain relationships [19]. Utilizing sentiment analysis to evaluate customer satisfaction is another example of understanding sentiments and emotions with natural language by taking into account the behavioral intention of customer's email expressing their negative sentiments. As a result, organizations can improve customer satisfaction by putting more efforts on the area that needs improvements such as service level agreement, accuracy and communication with the stakeholders [20].

2.4 Sentiment in Commit Logs

Sentiments have a massive impact on the productivity of developers. Developers' sentiments might fall in a wide range such as from being happy to sad or being frustrated to angry; hence resulting in delay in the product delivery or writing erroneous code [21]. Studies have been performed to understand software developer's sentiments using the commit message or log submitted on large software repositories like GIThub, SourceForge and many more. While working on the projects developers commit their code and with each commit they tend add a message to it. In the last decade, many researchers have collected such data and performed sentiment analysis to understand various relationships between the code quality and the sentiments of the developer. Now we will discuss some of the studies.

Guzman et al. [1] study the impact of emotions and sentiments on productivity, task quality and job satisfaction. The motivation behind the research was to understand the various factors involved that might affect the sentiments of the developers while working on projects like programming language used in the project, the day of the week the commit was made, the geographic location of the developer and how commit messages are related to the project approval [22]. A total of 90 top projects from Github which included 60,425 commit messages was collected to analyze sentiments using SentiStrength *“http://www.sentistrength.wlv.ac.uk/”* [23][24]. The research involved to understand if the emotions in commit messages or logs were having any relation with the programming language they were written into. The dataset involved 14 different programming languages, after performing analysis on the data the result was deduced that projects that have programming language as Java tend to have more negative comments as compared to any other programming language. It was also analyzed that the developer’s commit messages have different emotions based on the day of the week. 78% of the commit messages of the dataset used were written on weekdays where as 22% of the commit messages were written on weekends [25]. The data gathered was clustered together representing the number of commit messages made on each day of the week. It was perceived that Monday had the most negative commit comments. It was also analyzed on the basis of the geographic location, projects that have higher distribution lean to have more positive commit messages.

Over a period of time sentiment analysis has been playing a vital role to understand software developers mental state while writing the code but we also need to ensure to what extent these sentiment analysis softwares are accurate in terms of the emotions of software developers and also how much they agree with each other. Jongeling et al. [4] performed a study to understand the accuracy of sentiment analysis softwares with respect the emotions of the developers and to what extent these softwares agree with each other. They used SentiStrength, Alchemy, Stanford NLP sentiment analyzer and NLTK as sentiment analysis tools which takes commit messages as input and determine whether the commit message is a positive, negative or a neutral. Amongst all the tools mentioned SentiStrength has the most average accuracy [10]. It was found based on the analysis that NLTK and SentiStrength have a relation between the positive, negative and neutral values i.e. neutral sentiments are greater than negative sentiments and negative sentiments are greater than positive. It was observed that sentiment analysis in the field of software engineering have different outcome based on the tool selected for the study. Overall the tools do not agree with each other which lead to contradictory conclusions.

2.5 Discussion

In the previous section, we gave short descriptions of relevant studies conducted on sentiment analysis in the field of software engineering. There is an increasing amount of research in the software engineering community dealing with sentiment and the emotional aspect of software development. Numerous researchers have performed studies which determine how the texts written on microblogging websites, emails and commits made on the repositories relate to the sentiments of the person performing the activity. A number of tools have been developed to analyze these sentiments that give sentiment scores, which can be further grouped as positive, negative and neutral.

Through our research, we intent to determine how developers commit messages on Github repository relate to their respective sentiment. The work presented resembles the work done by Guzman et al. [1] with important differences such as we analyzed sentiments on commit messages or logs on much larger dataset – 2,251,585 observations. We used Boa *"http://boa.cs.iastate.edu/"* online querying system which contains Github dataset. We use this dataset as input and present a comprehensive analysis of all the 2,251,585 observations and their sentiment score. These observations were labeled into three different categories – Large, Medium and Small. In order to perform a detailed study and get more accurate results we used top five projects from each category Large (top five projects which contains maximum number of commit messages), Medium (top five projects which contains average number of commit messages) and Small (top five projects which contains low number of commit messages). In addition, we looked at the number of files changed and mapped it with the sentiments of the developers expressed in the commits that the files were part of, to determine if there was any relationship between the number of changed files in a commit and sentiment seen in commit logs. Finally, we used the commit logs and compared the output for all the three research questions using

SentiStrength and NLTK to analyze which sentiment analysis tool is providing us with more accurate results.

CHAPTER 3

METHODOLOGY AND EXPERIMENTS

This chapter presents the details of the various experiments we conducted on Boa infrastructure an online querying system, Python and SAS University Edition a statistical analysis tool used to perform extensive analysis.

3.1 Boa Infrastructure

In today's world ultra-large-scale software repositories such as SourceForge and GitHub are playing key role in the software engineering industry. They contain enormous amount of software development and related information which recently has been used by scientist, researchers, engineers and alike to analyze various aspects of software engineering industry [26][27]. But it is a challenging task to perform data mining on these repositories. Therefore, to address these challenges Boa was introduced. Boa is a domain specific language and infrastructure which has a collection of data from Github and Sourceforge repositories in order to perform data mining on these repositories [28]. We use the Boa infrastructure's online query system to generate commit logs as well as files that were changed during the commit to perform extensive analysis and discuss the research questions.

We used Boa Programming Guide [29] to understand the domain specific types which allows us to recognize the variables that we would require for the study. The data was collected by querying Boa system which involved a list of variables. Table 1 shows the list of variables, their type and description used from Boa.

Table 1: List of variables used from Boa [29]

	Attribute	Type	Description
Project	Id	string	Unique identifier for the project
	name	string	The name of the project
	developers	array of Person	A list of all software developers currently on the project
	programming_languages	array of string	A list of all programming languages used by the project
Revision	author	Person	The person who authored the revision, if known, otherwise the same as committer
	commit_date	time	The time the revision was committed
	committer	Person	The person who committed the revision
	files	array of ChangedFile	A list of all files committed in the revision
	id	int	A unique identifier for the revision
	log	string	The log message attached to the revision
ChangedFile	change	ChangeKind	The kind of change for this file
	kind	FileKind	The kind of file
	name	string	The full name and path of the file
enum ChangeKind	ADDED	Category	The file did not already exist and was added
	DELETED	Category	The file was deleted
	MODIFIED	Category	The file already existed and was modified
ASTRoot	imports	array of string	The imported namespaces and types

The "September 2015/GitHub(medium)" dataset was used from the Boa infrastructure as our data source. We divided the data extraction process into two different segments in order to reduce the runtime of the code (while extracting data from Boa) and the time taken to process the commit logs on the sentiment analysis tools (SentiStrength and NLTK). In addition, it also increases the efficiency of the code written in SAS which involves performing analysis only on selective columns. The output of the data extracted from Boa is saved as plain text. This file is imported into SAS in order to format it so that the columns are aligned and there are no discrepancies within the datasets.

3.2 Dataset

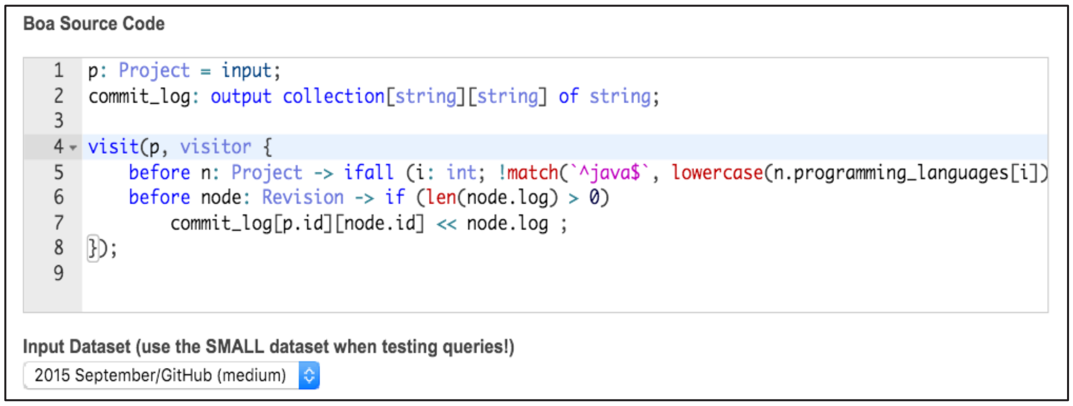
Figure 1 shows the program that is used to extract Project ID, Revision ID and the Revision Log. The program gives all non empty commit logs for all Java projects. Line 5 (shown below) of the program stops the processing of all projects that are not Java projects.

```
"before n: Project -> ifall (i: int; !match(`^java$`,
lowercase(n.programming_languages[i]))) stop;"
```

Line 6 (shown below) of the program filters only those records from the database which have a length greater than 0 for the commit log column.

```
"before node: Revision -> if (len(node.log) > 0)"
```

Similarly, Figure 2 shows the program which extracts Revision commit_date and Project name. We use the combination Project ID and Revision ID as primary key in order to join both the tables in SAS using proc sql procedures.



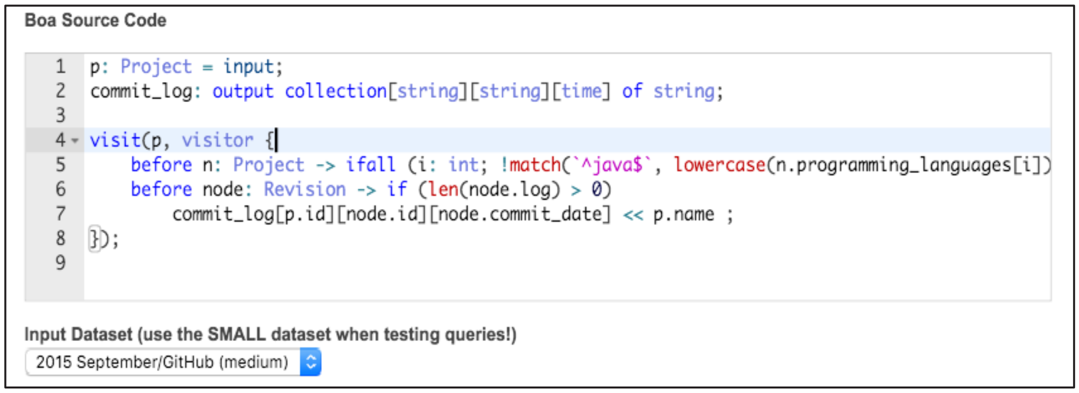
Boa Source Code

```
p: Project = input;
commit_log: output collection[string][string] of string;

visit(p, visitor {
    before n: Project -> ifall (i: int; !match(`^java$`, lowercase(n.programming_languages[i])
    before node: Revision -> if (len(node.log) > 0)
        commit_log[p.id][node.id] << node.log ;
});

```

Input Dataset (use the SMALL dataset when testing queries!)

2015 September/GitHub (medium)

Figure 1: Program to extract Project ID, Revision ID and Revision Log



Boa Source Code

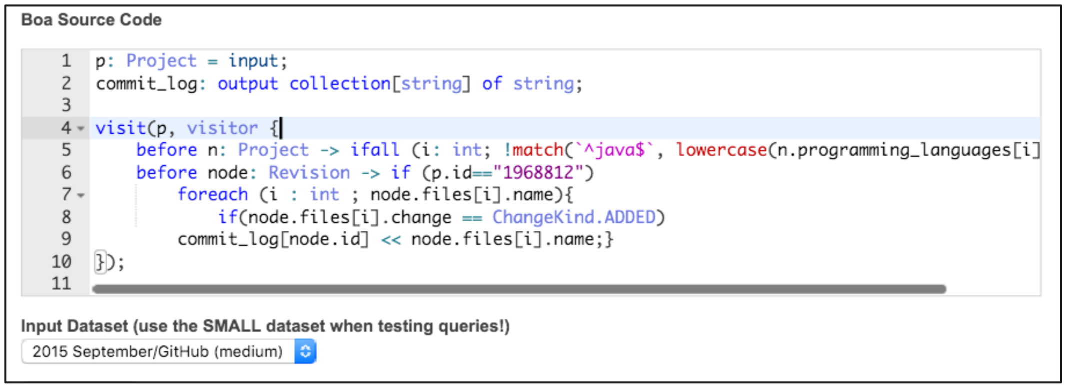
```
p: Project = input;
commit_log: output collection[string][string][time] of string;

visit(p, visitor {
    before n: Project -> ifall (i: int; !match(`^java$`, lowercase(n.programming_languages[i])
    before node: Revision -> if (len(node.log) > 0)
        commit_log[p.id][node.id][node.commit_date] << p.name ;
});

```

Input Dataset (use the SMALL dataset when testing queries!)

2015 September/GitHub (medium)

Figure 2: Program to extract Revision commit_date and Project Name

To perform analysis for RQ3 we extracted the data which comprises of the all changed files related to each commit. We were required to extract this data for fifteen different projects. The changed files fall into three different categories Added, Modified and Deleted. Therefore, the extraction process for changed file data had three different datasets for each project. Figure 3 shows the code snippet used to extract all the files data that were added with respect to each commit for a project. A unique id is generated with each commit and is termed as Revision ID, which is used to relate the number of files changed with each commit. Similarly, Figure 4 and Figure 5 shows the code snippet to extract all the changed files data that were modified and deleted respectively.



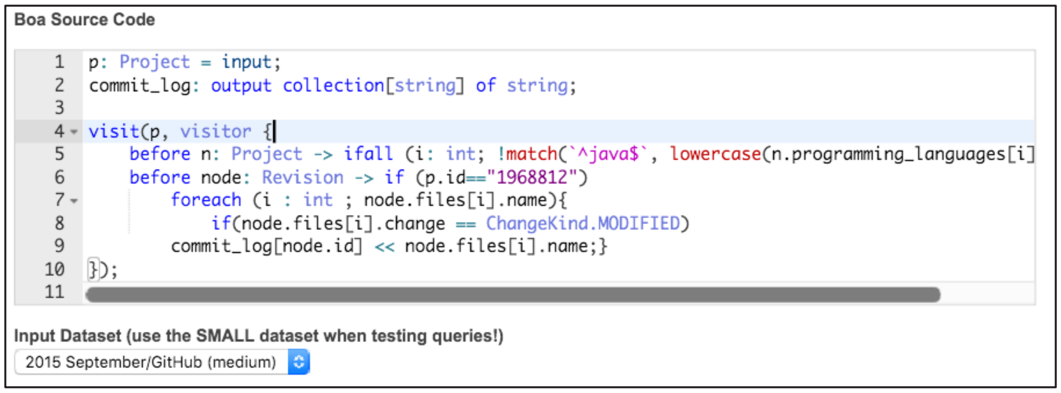
Boa Source Code

```
p: Project = input;
commit_log: output collection[string] of string;

visit(p, visitor {
    before n: Project -> ifall (i: int; !match(`^java$`, lowercase(n.programming_languages[i]
    before node: Revision -> if (p.id=="1968812")
        foreach (i : int ; node.files[i].name){
            if(node.files[i].change == ChangeKind.ADDED)
        commit_log[node.id] << node.files[i].name;}
});
```

Input Dataset (use the SMALL dataset when testing queries!)

2015 September/GitHub (medium)

Figure 3: Added files with respect to each commit



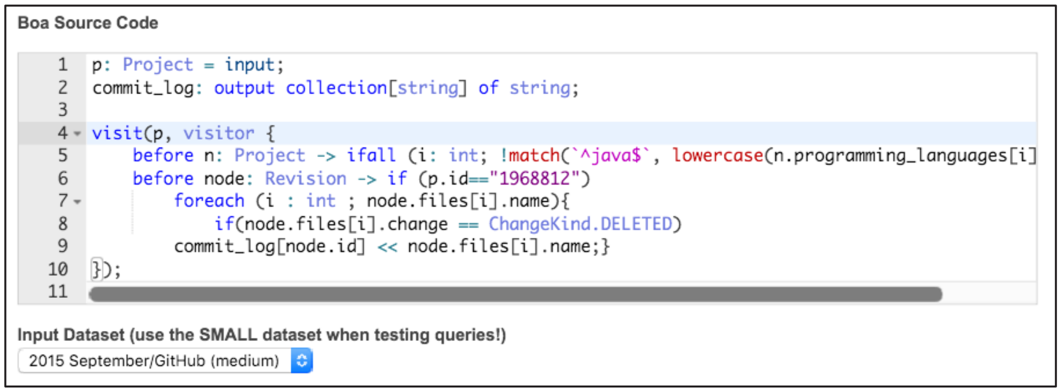
Boa Source Code

```
p: Project = input;
commit_log: output collection[string] of string;

visit(p, visitor {
    before n: Project -> ifall (i: int; !match(`^java$`, lowercase(n.programming_languages[i]
    before node: Revision -> if (p.id=="1968812")
        foreach (i : int ; node.files[i].name){
            if(node.files[i].change == ChangeKind.MODIFIED)
        commit_log[node.id] << node.files[i].name;}
});
```

Input Dataset (use the SMALL dataset when testing queries!)

2015 September/GitHub (medium)

Figure 4: Modified files with respect to each commit


Boa Source Code

```
p: Project = input;
commit_log: output collection[string] of string;

visit(p, visitor {
    before n: Project -> ifall (i: int; !match(`^java$`, lowercase(n.programming_languages[i]
    before node: Revision -> if (p.id=="1968812")
        foreach (i : int ; node.files[i].name){
            if(node.files[i].change == ChangeKind.DELETED)
        commit_log[node.id] << node.files[i].name;}
});

```

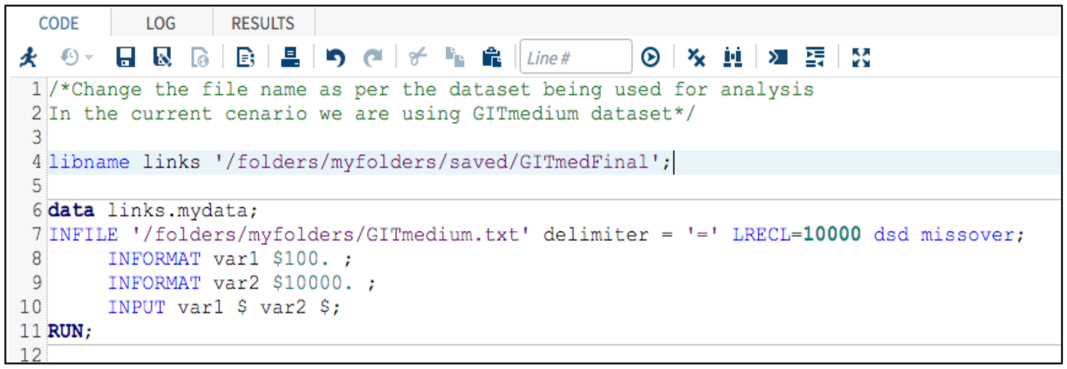
Input Dataset (use the SMALL dataset when testing queries!)

2015 September/GitHub (medium)

Figure 5: Deleted files with respect to each commit

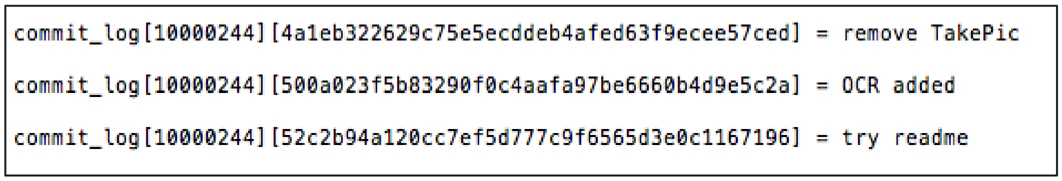
3.2.1 Data Cleaning and Formatting

The data extracted from Boa required to be formatted before processing it through the sentiment analysis tools SentiStrength and NLTK (using Python). Therefore, the data was imported into SAS; Figure 6 shows the code snippet to determine the import process of the raw file into SAS in order to format the data. We note here that we used SAS due to familiarity with the tool but this step could be done by any other means as well. By default, the data in SAS gets stored into work library, which is a temporary library which holds the data only for one SAS session. Once the session has ended the data gets erased automatically. We used the libname statement (Line 4) to store the data into a permanent library. Based on the output of Boa we have only two columns delimited by "=" as shown in Figure 7. The first column contains project id and revision id and the second column contains the commit log as shown in Figure 1.



```
/*Change the file name as per the dataset being used for analysis
In the current cenario we are using GITmedium dataset*/

libname links '/folders/myfolders/saved/GITmedFinal';

data links.mydata;
INFILE '/folders/myfolders/GITmedium.txt' delimiter = '=' LRECL=10000 dsd missover;
     INFORMAT var1 $100. ;
     INFORMAT var2 $10000. ;
     INPUT var1 $ var2 $;
RUN;
```

Figure 6: Boa Raw Data import process into SAS



```
commit_log[10000244][4a1eb322629c75e5ecddeb4afed63f9ecee57ced] = remove TakePic

commit_log[10000244][500a023f5b83290f0c4aafa97be6660b4d9e5c2a] = OCR added

commit_log[10000244][52c2b94a120cc7ef5d777c9f6565d3e0c1167196] = try readme
```

Figure 7: Boa Raw Data

Once the data gets imported into SAS we need to delete blank lines after each observation and perform certain preliminary checks in order to ensure that there are no discrepancies in the data. Figure 8 shows the code snippet to perform the data cleaning which involves deleting blank lines and perform validation checks so that we do not loose any data. Line 27 to 30 deletes the blank lines between each observation ensuring both columns are empty. We also observed that there were observations (commit logs) which contain long texts which jump into the next line first column as well after the import process. Line 32 to 52 aligns the data into appropriate columns and exports the data. The exported data is opened in Notepad++ to find “!@!” and replace it with blank which in return bring the long texts together. This is needed to join longer texts that span multiple lines.

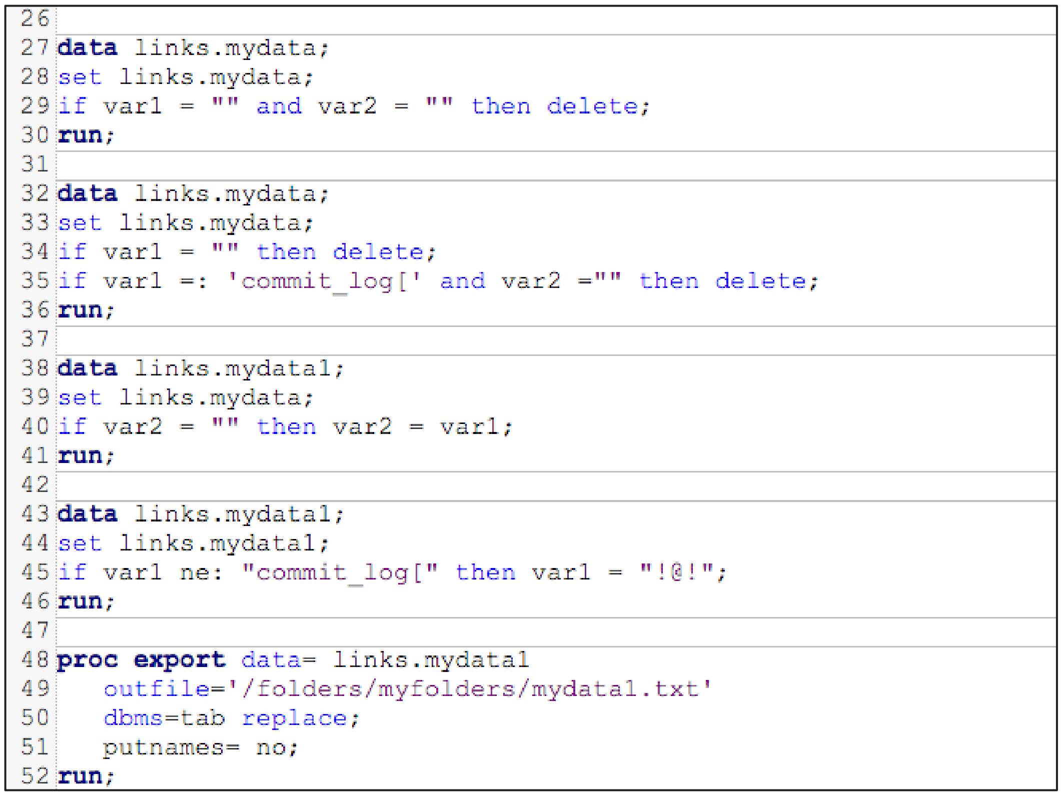

```
data links.mydata;
set links.mydata;
if var1 = "" and var2 = "" then delete;
run;

data links.mydata;
set links.mydata;
if var1 = "" then delete;
if var1 =: 'commit_log[' and var2 ="" then delete;
run;

data links.mydata1;
set links.mydata;
if var2 = "" then var2 = var1;
run;

data links.mydata1;
set links.mydata1;
if var1 ne: "commit_log[" then var1 = "!@!";
run;

proc export data= links.mydata1
   outfile='/folders/myfolders/mydata1.txt'
   dbms=tab replace;
   putnames= no;
run;
```

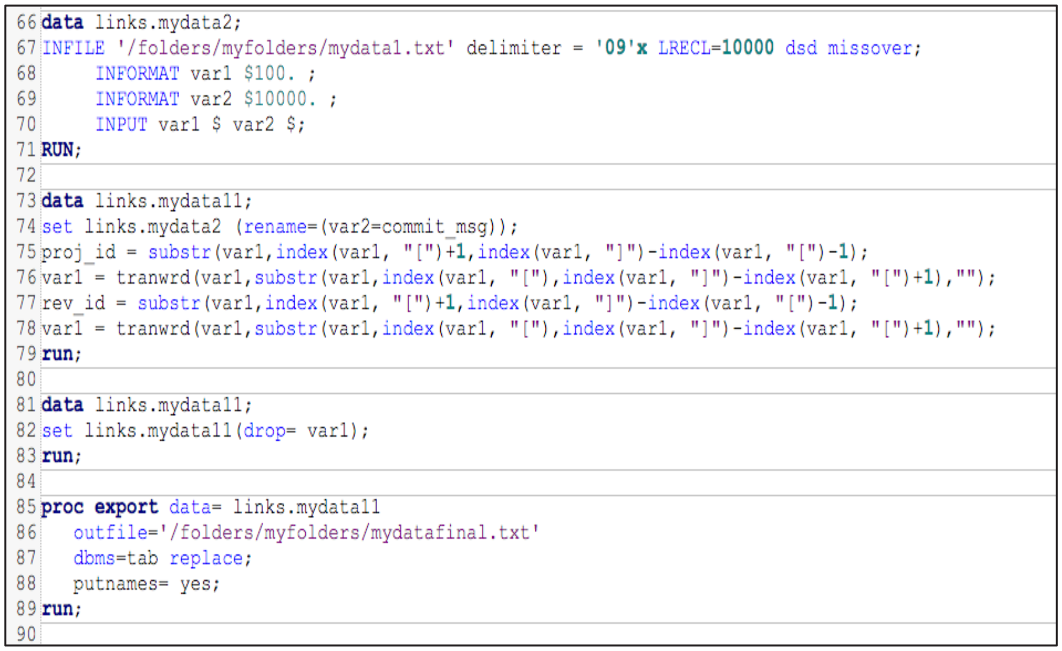
Figure 8: Preliminary checks

Once the data has been formatted it gets reimported in SAS. Figure 9 shows the code snippet to reimport the data. Column one still contains project id and revision id together which needs to be presented into two different columns. Therefore, to achieve this task three sas functions were used to extract the data: substr, index and tranwrd function [30].

- Substr: Extracts a substring from a string based upon start position and number of characters.

- Index: Finds the index number of a character in a string.
- Tranwrd: Replaces a substring within a string.

The data is finally exported as mydatafinal.txt to process it through sentiment analysis tools SentiStrength which requires a text file as input.

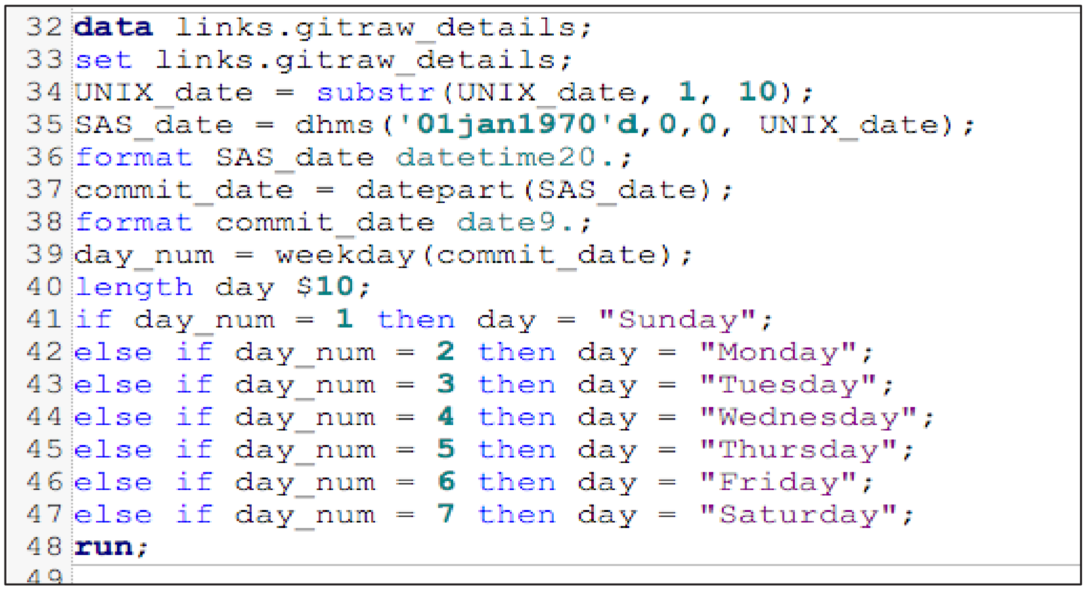


```
data links.mydata2;
INFILE '/folders/myfolders/mydata1.txt' delimiter = '09'x LRECL=10000 dsd missover;
     INFORMAT var1 $100. ;
     INFORMAT var2 $10000. ;
     INPUT var1 $ var2 $;
RUN;

data links.mydata11;
set links.mydata2 (rename=(var2=commit_msg));
proj_id = substr(var1,index(var1, "[")+1,index(var1, "]")-index(var1, "[")-1);
var1 = tranwrd(var1,substr(var1,index(var1, "["),index(var1, "]")-index(var1, "[")+1),"");
rev_id = substr(var1,index(var1, "[")+1,index(var1, "]")-index(var1, "[")-1);
var1 = tranwrd(var1,substr(var1,index(var1, "["),index(var1, "]")-index(var1, "[")+1),"");
run;

data links.mydata11;
set links.mydata11(drop= var1);
run;

proc export data= links.mydata11
   outfile='/folders/myfolders/mydatafinal.txt'
   dbms=tab replace;
   putnames= yes;
run;

```

Figure 9: Reimporting and Assigning Column name

3.2.2 Data Preprocessing – Commit Date

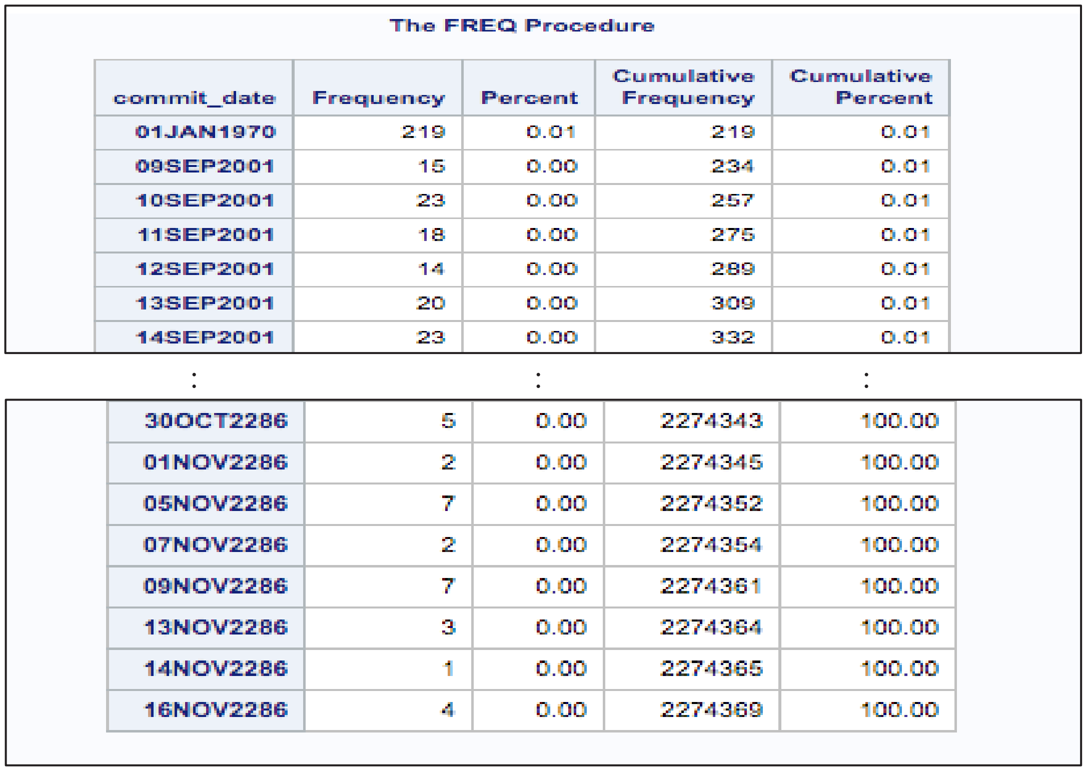
In order to perform analysis for RQ 2, finding the relationship between sentiment score and the day of the week, it was required to join the two tables exported from Boa. The commit_date extracted from the Boa query was in UNIX format (datetime). Therefore, the date was converted into SAS date format (datetime20.) and further date and time were separated using the datepart function. SAS inbuilt function weekday was used to find the day of the week the commit was made. Figure 10 shows the code snippet to get SAS date and determine the day of the week from UNIX date format.



```
data links.gitraw_details;
set links.gitraw_details;
UNIX_date = substr(UNIX_date, 1, 10);
SAS_date = dhms('01jan1970'd,0,0, UNIX_date);
format SAS_date datetime20.;
commit_date = datepart(SAS_date);
format commit_date date9.;
day_num = weekday(commit_date);
length day $10;
if day_num = 1 then day = "Sunday";
else if day_num = 2 then day = "Monday";
else if day_num = 3 then day = "Tuesday";
else if day_num = 4 then day = "Wednesday";
else if day_num = 5 then day = "Thursday";
else if day_num = 6 then day = "Friday";
else if day_num = 7 then day = "Saturday";
run;
```

Figure 10: UNIX date to SAS date format

It was observed after performing a frequency check on the commit_date extracted (using Boa query as shown in Figure 2 and further processing it into SAS as shown in Figure 10 to get SAS date) that there are discrepancies in the date attached to each commit log. Figure 11 shows samples of these data discrepancies.

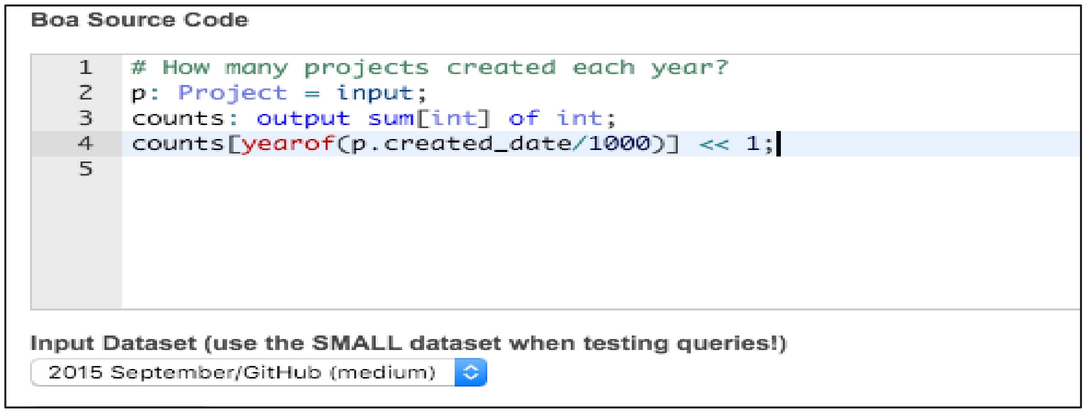


The FREQ Procedure

commit_date	Frequency	Percent	Cumulative Frequency	Cumulative Percent
01JAN1970	219	0.01	219	0.01
09SEP2001	15	0.00	234	0.01
10SEP2001	23	0.00	257	0.01
11SEP2001	18	0.00	275	0.01
12SEP2001	14	0.00	289	0.01
13SEP2001	20	0.00	309	0.01
14SEP2001	23	0.00	332	0.01
:		:		:
30OCT2286	5	0.00	2274343	100.00
01NOV2286	2	0.00	2274345	100.00
05NOV2286	7	0.00	2274352	100.00
07NOV2286	2	0.00	2274354	100.00
09NOV2286	7	0.00	2274361	100.00
13NOV2286	3	0.00	2274364	100.00
14NOV2286	1	0.00	2274365	100.00
16NOV2286	4	0.00	2274369	100.00

Figure 11: Data discrepancies in Commit date

To have a clean dataset for RQ 2 we came across a solution. The first step involved querying Boa to get the number of projects created each year.



Boa Source Code

```
# How many projects created each year?
p: Project = input;
counts: output sum[int] of int;
counts[yearof(p.created_date/1000)] << 1;
```

Input Dataset (use the SMALL dataset when testing queries!)

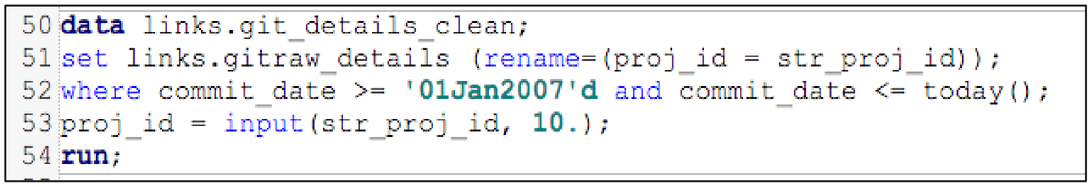
2015 September/GitHub (medium)

Figure 12: Projects created each year

Table 2: Number of projects created each year

Year	# Projects Created
2007	2
2008	3059
2009	15311
2010	31600
2011	86179
2012	231019
2013	416812

Figure 12 shows the code used to get the count of projects created each year and Table 2 shows the output. Any Commit date cannot be before the project created date and it was concluded that to analyze RQ 2 we need need to remove all the date before 1st Jan 2007. For the end date, we used the today function which takes all the commit messages till 2016 and removes the rest. Figure 13 shows the code used to remove the incorrect commit_dates from the dataset. After this process, we had 2,130,474 observations which we used for further analysis. This data was linked with the table having the sentiment score to get the sentiment polarity for each day of the week.



```
data links.git_details_clean;
set links.gitraw_details (rename=(proj_id = str_proj_id));
where commit_date >= '01Jan2007'd and commit_date <= today();
proj_id = input(str_proj_id, 10.);
run;
```

Figure 13: Removal process for incorrect commit_date

To perform in depth analysis, the data was split into three different categories to have a better understanding of these sentiments and their relationships on a project level. Data having the maximum number of commit messages or logs, data having average number of commit messages and data having low number of commit messages. This bifurcation was done specifically to understand and analyze RQ 2 and RQ 3. We considered top five projects from all the three different categories:

- **Dataset 1:** Table 3 shows project information for the top five projects having maximum number of commit logs.
- **Dataset 2:** Table 4 shows project information for the top five projects having average number of commit logs.
- **Dataset 3:** Table 5 shows project information for the top five projects having low number of commit logs.

Table 3: Top 5 projects with **Maximum** number of commit logs

Proj_id	Proj_name	Count(commits)
1968812	dava/dava.framework	14969
7785050	fieldtrip/fieldtrip	10502
5153143	cwi-swat/rascal	9847
10613094	codefireXperiment/libco	9793
12496978	Droid-Concepts/packages	9360

Table 4: Top 5 projects with **Average** number of commit logs

Proj_id	Proj_name	Count(commits)
13010741	houst0nn/android_packag	4746
6719407	iGio90/android_packages	4726
365893	justinedelson/felix	4610
2424377	carrot2/carrot2	4574
5256179	nourlcn/yarn-comment	4584

Table 5: Top 5 projects with **Low** number of commit logs

Proj_id	Proj_name	Count(commits)
1571039	echo3/echo3extras	1254
4067771	pixelballoon/pixelboost	1251
11416657	madejson/SpolecznoscFin	1243
10530838	Distrotech/shared-mime-	1242
10453653	ema/conpaas	1235

3.3 Experiments Conducted

The analysis was performed using two tools SentiStrength and NLTK (a Python package) on the dataset extracted as discussed above.

3.3.1 SentiStrength – Sentiment Analysis Tool

To determine the sentiment polarity developers convey while submitting code revisions and commit logs, we use the sentiment analysis tool SentiStrength. This tool was chosen because of the high accuracy rates reported in previous studies on Twitter data. SentiStrength was also used in software engineering studies. For example Guzman et al. [1] investigated the relationship between sentiment in commit messages and the programming language used, the day of the week when the commit was submitted and the overall project approval. First, SentiStrength tokenizes the text and then assigns a score for each word that conveys an emotion. Words with negative implications are given scores between -1 to -5 and words with positive sentiment values receive an integer value between 1 and 5. Next, the word's scores for each commit log are summarized to generate a pair of values known as the senti score. The first value in the senti score indicates the positive score and second value is the negative score for the sentence. Let us illustrate this with an example. Consider the following commit log: "Added basic flying monster animation in project" taken from project ID "`10002651`". After analyzing each word SentiStrength provides the scores for each word and the sentence as

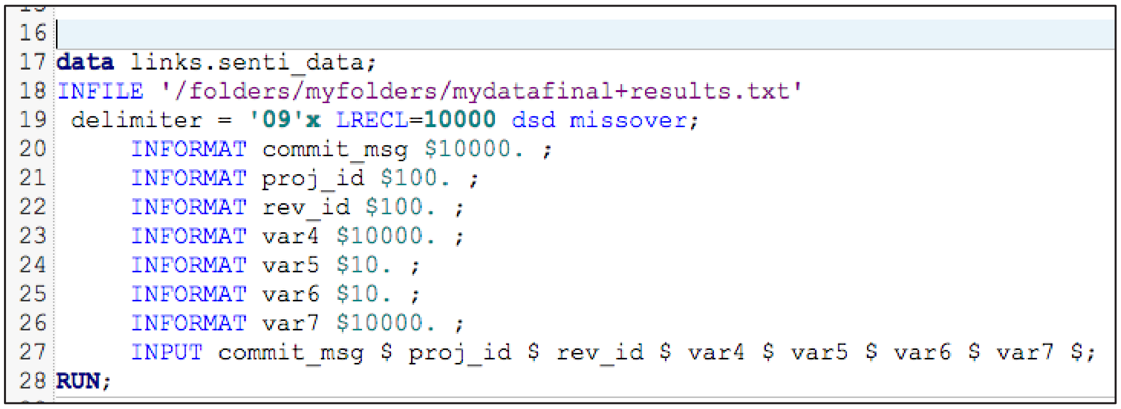
"`Added[0] basic[0] flying[0] monster[-1] animation[0] [[Sentence=-2,1=word max, 1-5]][[[1,-2 max of sentences]]]`". Note the senti score of [1,-2] where 1 indicates the maximum positive score for the sentence and -2 is the maximum negative score for the sentence.

To find the final sentiment score for a commit log, we take the sum of the maximum positive and the maximum negative score given by the senti score. The final sentiment score is used to find the overall commit log polarity sentiment as positive, neutral or negative. In the above example, the final sentiment score would be -1 (sum of 1 and -2). A positive sum represents a positive sentiment, zero represents a neutral sentiment or no emotion, and a negative sum indicates a negative sentiment. Table 6 shows examples from all three categories of commits from our GitHub dataset and the final score that is a sum of the senti score.

Table 6: Sample commit messages categorized by sentiment polarity

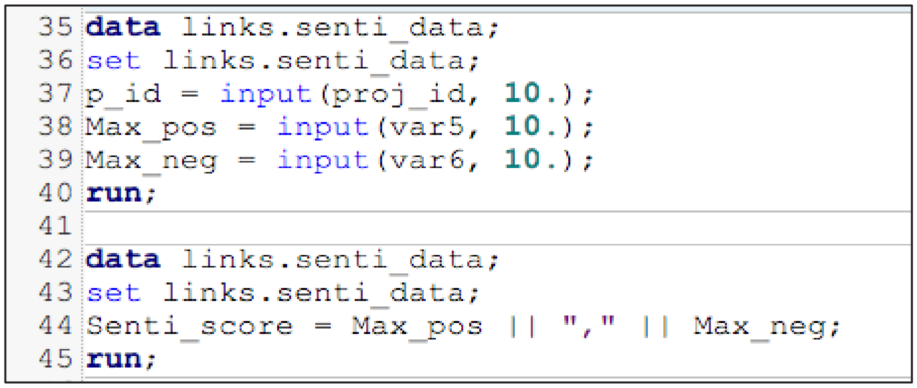
Sentiment	Commit Message	Final Score
Positive	We're not totally terrible.	4
	Build success !!!	3
	pretty pretty code	3
	Added parallelism and seems it works fine :)	3
	A few finishing touches that Anna liked :)	3
	Small tweaks on top of Daniel's excellent refactoring git-svn-id	3
Negative	Terrible, terrible mock folder guid retrieval.	-4
	Trying to complete the qualifier 3. Grounds for suicide :(	-4
	Fix heinous TMemoryBuffer bug and warning in FileTransport Review	-4
	Attempted to fix map camera failed horribly	-4
	ENH: very painfully merge: svn merge --accept	-4
Neutral	initial commit Committer: Jeremy Truelove jtruelove@gmail.com	0

The data that was exported as shown in Figure 9 was processed through the SentiStrength tool to get the sentiment score for each commit message. The output file from SentiStrength was imported into SAS for further analysis. Figure 14 shows the code snippet used to import the output from SentiStrength into SAS. As SentiStrength gives two columns which are used to calculate the final score so we label them as “`Max_pos`” and “`Max_neg`” which gives the maximum positive score for the sentence and the maximum negative sentence respectively. Figure 15 shows the code snippet used to label the columns as “`Max_pos`” and “`Max_neg`”, in addition it also adds a new column to which is a combination of Max_pos and Max_neg score in a format which is a standard SentiStrength score format as discussed above.



```
data links.senti_data;
INFILE '/folders/myfolders/mydatafinal+results.txt'
 delimiter = '09'x LRECL=10000 dsd missover;
     INFORMAT commit_msg $10000. ;
     INFORMAT proj_id $100. ;
     INFORMAT rev_id $100. ;
     INFORMAT var4 $10000. ;
     INFORMAT var5 $10. ;
     INFORMAT var6 $10. ;
     INFORMAT var7 $10000. ;
     INPUT commit_msg $ proj_id $ rev_id $ var4 $ var5 $ var6 $ var7 $;
RUN;
```

Figure 14: SentiStrength output import into SAS



```
data links.senti_data;
set links.senti_data;
p_id = input(proj_id, 10.);
Max_pos = input(var5, 10.);
Max_neg = input(var6, 10.);
run;

data links.senti_data;
set links.senti_data;
Senti_score = Max_pos || "," || Max_neg;
run;
```

Figure 15: Column Label and SentiStrength output format

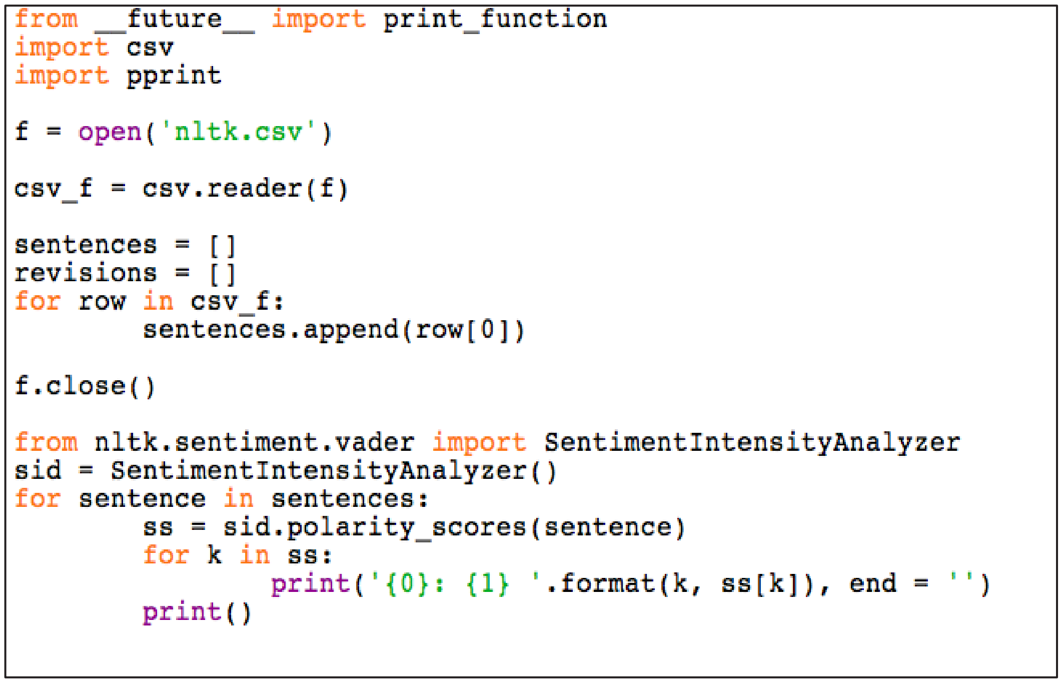
3.3.2 NLTK – Sentiment Analysis using Python

In order to perform analysis for RQ 4 it was required to compare results for two different sentiment analysis tools. Hence, we also used NLTK, a library that can be imported into Python to perform sentiment analysis on texts. We used the same formatted data (commit messages) from SAS to process them through NLTK. When a text is processed through NLTK, it determines the probable emotion of the sentiment. As an output it returns the probability of being negative, one probability of being neutral and one probability of being positive. The probability score for each category determines if the text is neutral, negative or positive. If the probability score for neutral is greater than 0.5, the text is considered neutral. Otherwise, it is considered to be the other sentiment with the highest probability [4][31]. We used the Vader package [32][33] from NLTK to perform the sentiment analysis on 2,251,585 observations (exported from SAS as shown in Figure 9) using Python. Let us determine this with a help of an example. Consider the following commit log:

`"photos working with photo intent, problem was cyanogenmod"`

The above mentioned commit log is taken from project ID “`10000244`” and revision ID “`9ccab82fa3cc8a970e2fb7cc628e0e4848ca2be9`”. After analyzing the text the Vader package from NLTK determines one probability of the text being neutral, one probability of text being negative and one probability of text being positive. In addition the Vader package also provides a compound value which lies between -1 to 1, 0 being neutral. This compound value is used to determine if the text is neutral, negative or positive. The output of the text processed with NLTK is in the following format “`{neg: 0.278 neu: 0.722 pos: 0.0 compound: -0.4019}`”. Note that the compound value is negative, therefore, the entire text sentiment polarity will be negative.

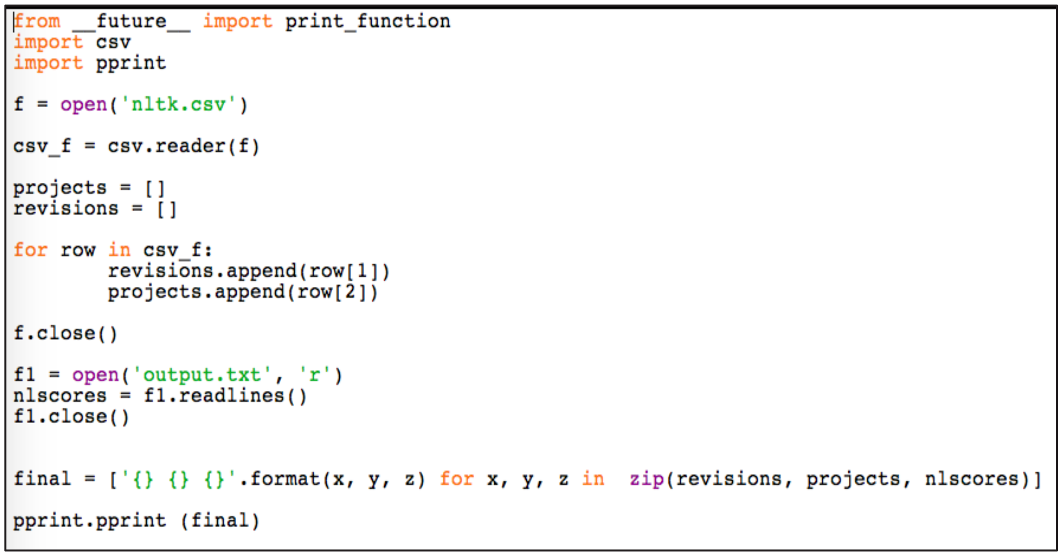
The data from SAS, which includes commit log, project id and revision id was exported in .csv format. Figure 16 shows the script written in Python to import the data. Further the process involves selecting all commit logs in to one single list to calculate the sentiment score. The Python script was executed in the terminal window using the following command "`python nltk_senti_1.py >> output.txt`". This command executes the Python script and exports the data into a text file.



```
from __future__ import print_function
import csv
import pprint

f = open('nltk.csv')

csv_f = csv.reader(f)

sentences = []
revisions = []
for row in csv_f:
        sentences.append(row[0])

f.close()

from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
for sentence in sentences:
        ss = sid.polarity_scores(sentence)
        for k in ss:
                print('{0}: {1} '.format(k, ss[k]), end = '')
        print()
```

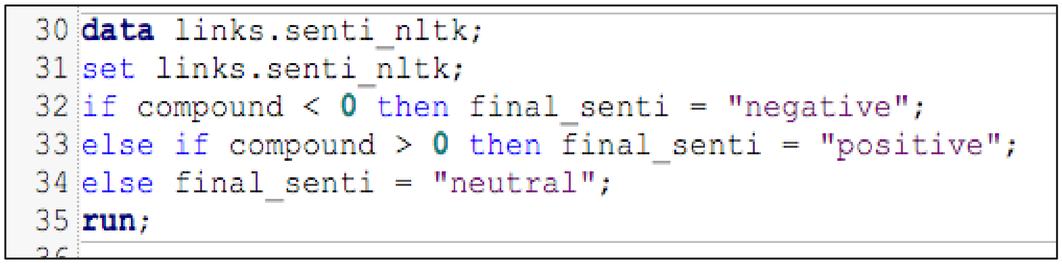
Figure 16: Processing Commit Logs through NLTK

After the sentiment score was obtained, the next process involved linking the NLTK sentiment score with the project id and revision id in order to reimport them into SAS and perform further analysis as we performed it for SentiStrength. Figure 17 shows the Python script to link the sentiment score obtained by executing the script shown in Figure 16, revision id and project id. The Python script was executed in the terminal window using the following command "`python nltk_senti_2.py >> final.txt`". The output was imported in SAS and a new column was added to the dataset which determines the sentiment polarity of the developer in each commit message or log. Figure 18 shows the code snippet to calculate the sentiment polarity on the basis of the compound value. If the compound value is less than zero the commit log has a negative sentiment. If the compound value is greater than zero the commit log has a positive sentiment and finally, if the compound value is zero the commit log has a neutral sentiment.



```
from __future__ import print_function
import csv
import pprint

f = open('nltk.csv')

csv_f = csv.reader(f)

projects = []
revisions = []

for row in csv_f:
        revisions.append(row[1])
        projects.append(row[2])

f.close()

f1 = open('output.txt', 'r')
nlscores = f1.readlines()
f1.close()


final = ['{} {} {}'.format(x, y, z) for x, y, z in  zip(revisions, projects, nlscores)]

pprint.pprint (final)
```

Figure 17: Process to link NLTK sentiment score, project id and revision id



```
data links.senti_nltk;
set links.senti_nltk;
if compound < 0 then final_senti = "negative";
else if compound > 0 then final_senti = "positive";
else final_senti = "neutral";
run;
```

Figure 18: Script to calculate the Sentiment polarity for each commit log

CHAPTER 4

RESULTS AND ANALYSES

This chapter presents the results and determines the relationship of the commit log sentiments and with the various aspects discussed in this thesis.

4.1 RQ1: What is the general developer sentiment in commit messages for GitHub projects?

Table 7 shows the frequency of sentiment score after processing 2,251,585 commit logs on SentiStrength. To calculate the final score, we sum the both values of the sentiment score. The results are clustered together on the basis of final score for each commit logs as shown in Table 8. We notice that 74.74% of the commits had a neutral sentiment, 7.19% had a positive sentiment and 18.05% had a negative sentiment. This indicates that there were more than twice the number of negative sentiment commit logs compared to positive sentiment commit logs. We notice that a maximum number of the commits fall into the senti score range of [1,-1]. This could happen because many of the commit messages have very neutral commit messages such as: Added file, changed description, etc. In addition to that a number of developers tend to add URLs and/or variable names and/or a piece of code which they modify in commit messages rendering them as neutral. This is also indication that further work is needed to adapt sentiment analysis tools to software engineering artifacts.

Table 7: Frequency for each Sentiment Score

Senti_score	COUNT
1, -1	1612199
1, -2	353873
1, -3	38843
1, -4	2756
1, -5	66
2, -1	142841
2, -2	68281
2, -3	9793
2, -4	923
2, -5	37
3, -1	11487
3, -2	7037
3, -3	2515
3, -4	187
3, -5	4
4, -1	367
4, -2	294
4, -3	53
4, -4	14
5, -1	10
5, -2	4
5, -3	1

Table 8: Sentiments across all Commits

Sentiment	Final Sentiment Score	Number of Commits	Sentiment Percentage
Negative	-4	66	18.05%
	-3	2793	
	-2	39770	
	-1	363853	
Neutral	0	1683009	74.75%
Positive	1	149931	7.20%
	2	11782	
	3	371	
	4	10	
	Total	**2251585**	

To analyze this further, we split the dataset into three subsets: large, average, and low number of commits and considered only the top five projects in each of these categories as discussed in section 3.2.2. Table 9 shows the three split datasets for further analysis. The split was done manually after looking at a sorted distribution of commits in the projects. A total of 83,936 commits were part of the subset analysis.

Table 9: Top five projects from the subset of data categorized into large, average, and low number of commits

Data Subset	Total # Commits	Min Commits	Max Commits	Mean Commits
Large	54471	9360	14969	10894.2
Average	23240	4574	4746	4648
Low	6225	1235	1254	1245

We report the results of the sentiment score in Table 10 after running SentiStrength on these subsets. We notice similar trends in these subsets when we compare them to Table 8. The only difference is that for projects with low number of commits, the positive and negative sentiment seem to fall closer together (only 5% apart) whereas in projects with large and average number of commits, the negative sentiment is on average 14% higher than the positive sentiment. Figure 19 shows the distribution of commit messages for each sentiment score. The large and average number of commits follow the exact same trend but neutral score "0" for low number of commits was approximately 11% apart from large and average number of commits. This could be because as the project progresses (with more commits), it gets more complex involving more developers and thus more issues arise causing sentiment to move towards the negative direction. This also does not necessarily mean that the project is not productive or of good quality. Another set of analysis needs to be conducted to determine code quality, which in turn needs to be mapped to the sentiment analysis done here. It was also observed in Figure 19 for final score "-1", the low dataset is approximately 9% apart from the large and the average datasets.

Table 10: Number of Commits

Sentiment	Large	Average	Low
Negative	21.14%	22.33%	11.49%
Neutral	71.05%	70.45%	82.47%
Positive	7.81%	7.22%	6.04%

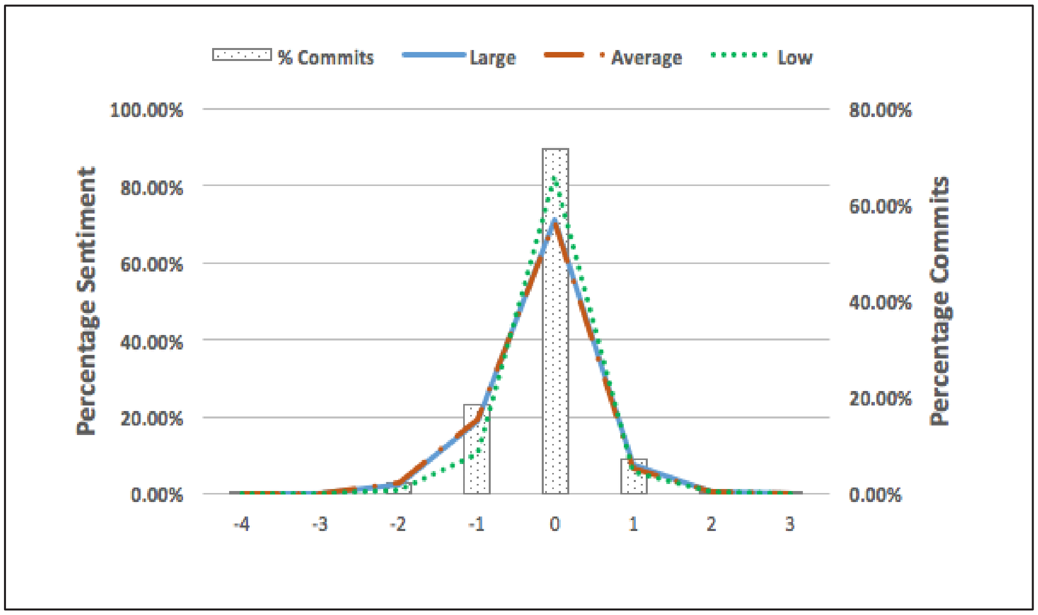


Figure 19: Sentiment Score Distribution

One reason for the high neutral sentiment could be because commits in general are different than tweets or online reviews. When people write reviews their goal is to express feelings of satisfaction or dissatisfaction about a product or movie. Software developers write commit logs anytime a revision is submitted to a software repository, therefore most of the time there is no human emotion or sentiment involved. The small percentage of commits that exhibit a positive or a negative sentiment polarity show different types of emotions than other types of online postings [3].

4.2 RQ2: What is the relationship between developer sentiment in commit messages and the day of the week the commit was made?

In this research question, we wanted to determine if the day of the week plays a role in developer sentiment for commit logs. As discussed in section 3.2.2, we removed all commits with a commit date before their project's creation date and all commits with a date in the future. All the projects that remained were created between 2007 and 2013. There were 2 projects created in 2007 and 416,812 projects created in 2013 as shown in Table 2. Based on the commit date, we calculated the day of the week the commit was made and grouped sentiment by day.

For this analysis, we choose to look at two representative scores given by SentiStrength namely, the maximum positive with the least negative sentiment (MAX +ve) and the maximum negative with the least positive sentiment (MAX –ve). We believe these two extremes are more important as a lot of the sentiment gets averaged out if scores are added together. Figure 1 shows the percentage of these sentiment scores across the day of week along with the percentage of commits done on that day. It can be seen that most commits were done on Wednesday, which also sees the second highest maximum positive and maximum negative sentiment. The highest maximum negative sentiment is seen on Tuesday across all projects. Considering only weekdays, the lowest positive sentiment and the lowest negative sentiment are both seen on Mondays with 5% more positive sentiment.

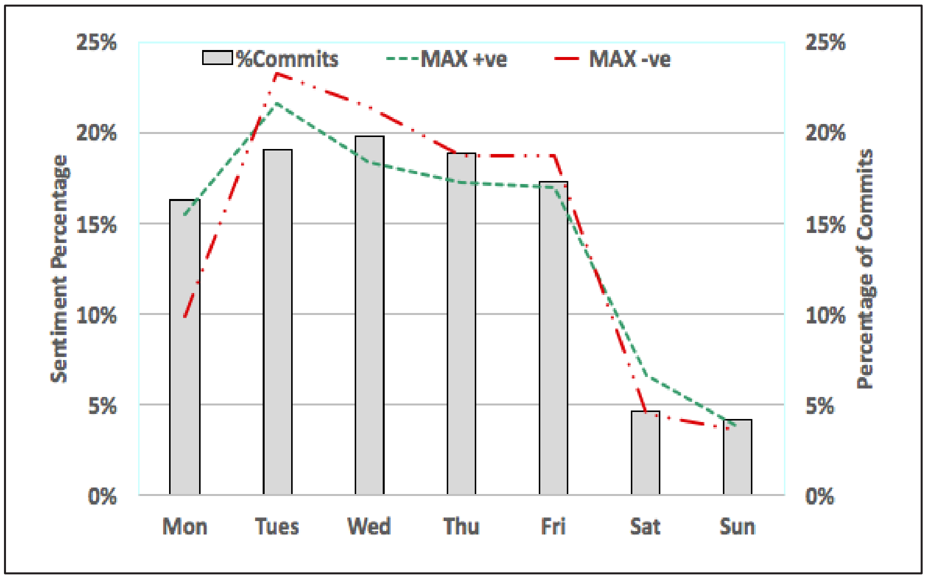


Figure 20: Maximum positive and maximum negative sentiment across all projects with respect to day of week

To perform in depth analysis, we further breakdown of these percentages across three categories of commits (Large, Average and Low) as shown in Table 9. Figure 21 projects with the most number of commits. We see the highest negative sentiment on Wednesday and Thursday. Thursday also had the highest positive sentiment (and lowest negative sentiment). In the large subset, SentiStrength only had values for Wednesday and Thursday for the maximum negative sentiment. Hence, this group of large committers do not follow the average of all projects as shown in Figure 1 where in general Tuesday was the most negative day. Figure 22 shows that the projects with average number of commits had Tuesday as the most negative but we also find that it had the most positive sentiment. Finally, Tuesday was again the day with the most negative sentiment and Fridays had the most positive sentiment for the projects with fewer commits as shown in Figure 23. To conclude RQ2 findings, we find that there are trends in sentiment across the days of the week that differ based on the project's number of commits.

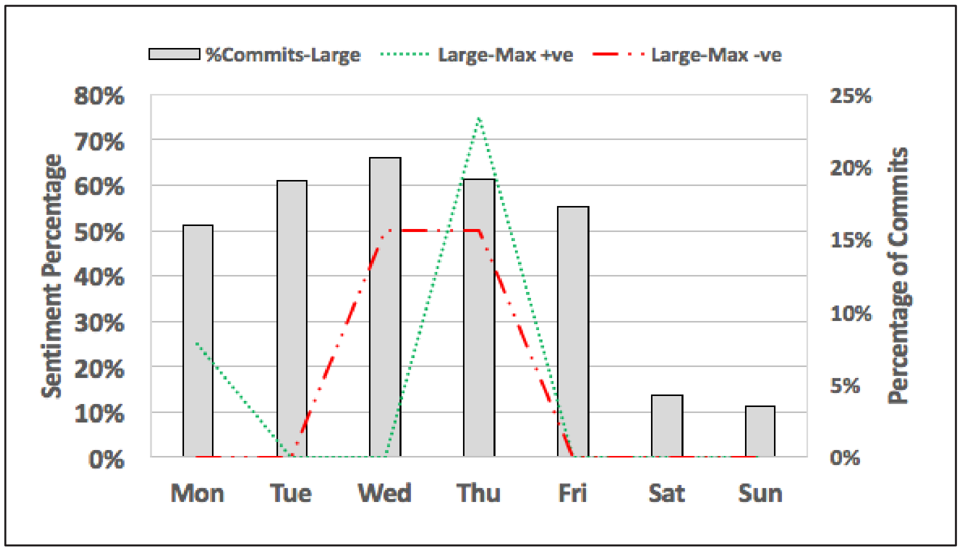


Figure 21: Maximum positive and maximum negative sentiment for projects having maximum number of commits with respect to day of week

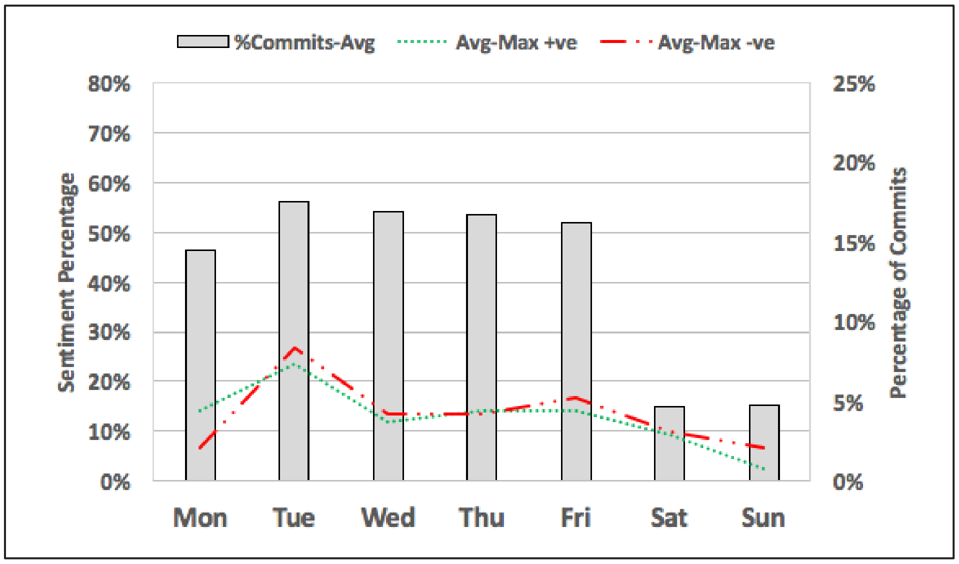


Figure 22: Maximum positive and maximum negative sentiment for projects having average number of commits with respect to day of week

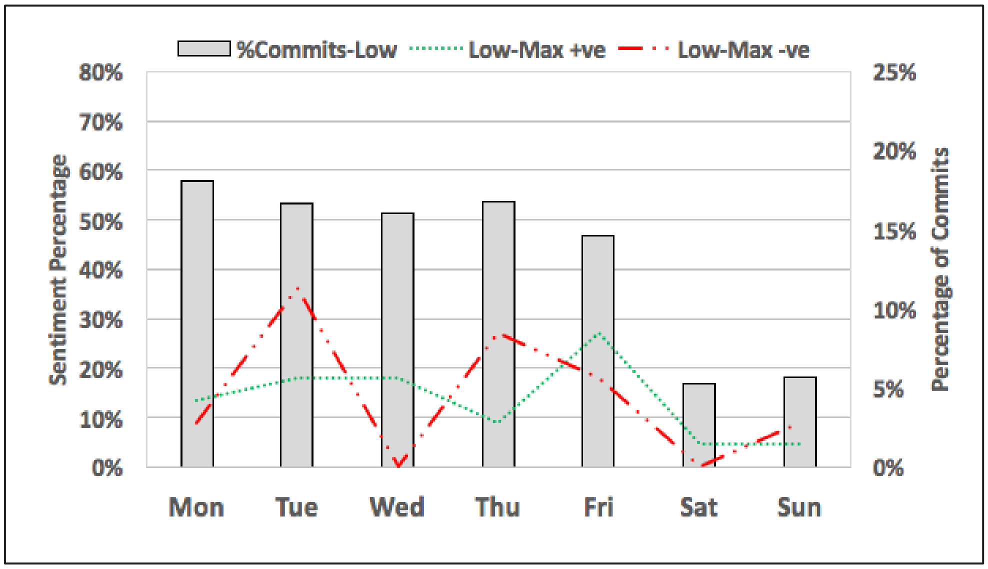


Figure 23: Maximum positive and maximum negative sentiment for projects having low number of commits with respect to day of week

4.3 RQ3: Is there a correlation between the number of changed files and developer sentiment?

In RQ3, we wanted to determine if there was any relationship between the number of changed files in a commit and sentiment seen in commit logs. To do this, we queried Boa to give us all the files that were added, modified, and deleted across the top five projects in each of the large, average, and low commit categories as shown in Figure 3, Figure 4, and Figure 5 respectively. The number of files changed is the sum of all the files that were added, modified, and deleted with each commit. Figure 24 shows the overall result for all projects. We group together final scores of positive, negative, and neutral sentiment to show how sentiment changes across time along with the number of files changed. The number of files changed (line graphs) in a commit mapped to each sentiment is shown on the Y- axis to the right. The Y-axis on the left denotes the average number of changed files per commit (bar graph) during the year.

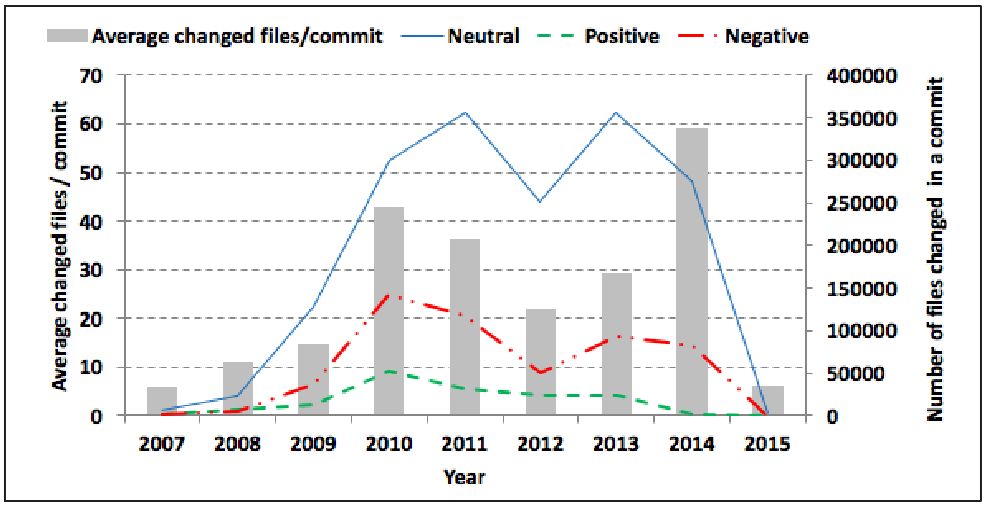


Figure 24: Positive, negative and neutral sentiments across all projects with respect to average changed files per commit

For the large subset of the top 5 projects as shown in Figure 25, we notice that in the year 2014 there was a maximum number of changed files per commit (~60.35). We do notice a spike in the negative sentiment at this time as well (see 2010 for similar trend). There is a spike in positive sentiment too but not as prominent as the negative sentiment. In the average subset, we find 2009 and 2011 to be the most negative overall. Figure 26 shows the result. The year 2011 also has the highest number of changed files. However, we also see a decrease in negative sentiment in Year 2010 from 2009 even though the number of files changes was higher in 2010. In the low category of commits, the negative and positive sentiment are almost the same across all the years. An unusual spike in neutral sentiment in the year 2014 was observed as shown in Figure 27. The maximum number of files changed was in the year 2012 which caused the positive and the negative sentiment to spike slightly in the following year.

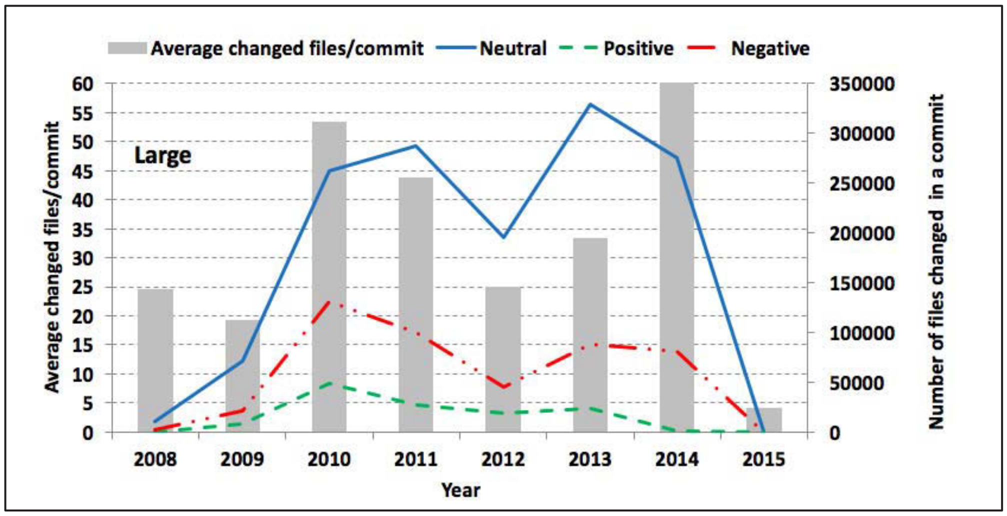


Figure 25: Positive, negative and neutral sentiments for projects having maximum number of commits with respect to average changed files per commit

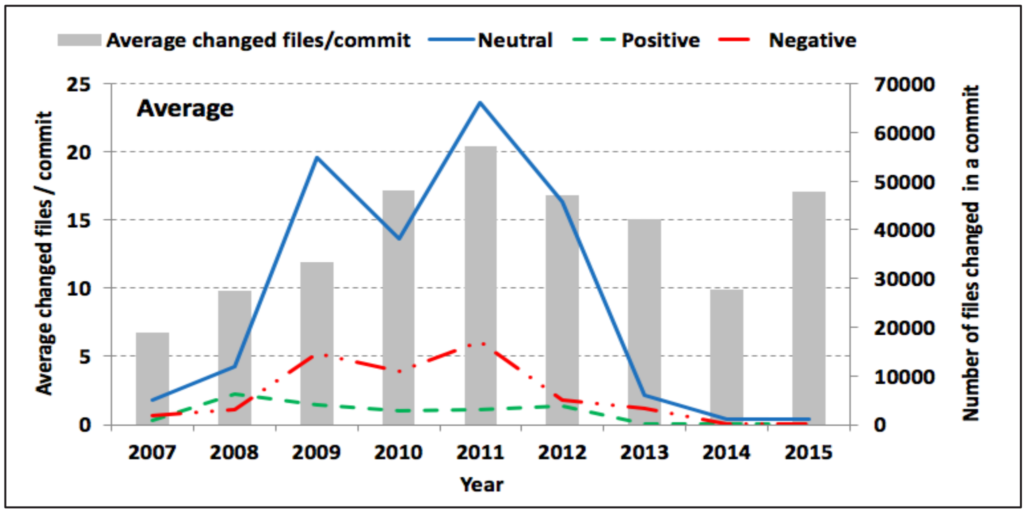


Figure 26: Positive, negative and neutral sentiments for projects having average number of commits with respect to average changed files per commit

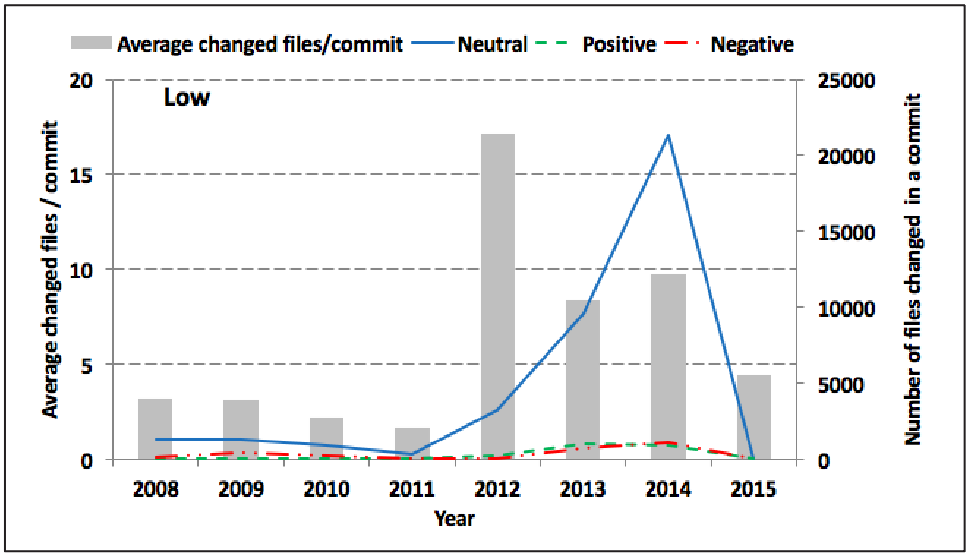


Figure 27: Positive, negative and neutral sentiments for projects having low number of commits with respect to average changed files per commit

We found strong correlations (using Pearson's correlation test > 0.95) between the negative, positive, and neutral number of commits and the average number of changed files especially for the large and low subsets but not the average subset.

4.4 RQ4: How do existing sentiment analysis tools compare when applied to commit messages?

We used SentiStrength for RQ 1, RQ 2 and RQ 3. For RQ 4, we wanted to compare the results of SentiStrength with another sentiment analysis tool NLTK as discussed in section 3.3.2. All the 2,251,585 commit logs were processed in Python and the sentiment was calculated using the Python package *"nltk.sentiment.vader"*. Table 11 shows a comparison for overall developer sentiment between SentiStrength and NLTK. The maximum difference was observed in positive sentiment (15.21%). The neutral and the negative sentiment has 9.56% and 5.65% difference respectively.

Table 11: Sentiment Comparison on the across all Commits

Sentiment	% SentiStrength	% NLTK
Negative	18.05%	12.49%
Neutral	74.75%	65.10%
Positive	7.20%	22.41%

We report the results of the sentiment score in Table 12 after running NLTK on these subsets (large, average and low). Figure 28 shows difference between the positive, neutral and negative sentiment after comparing SentiStrength (Table 1) and NLTK (Table 12). It was observed that the projects having the average number of commits have the maximum variance for all the three categories (positive, neutral and negative). Projects having low number of commits have less variance when compared with projects having large and average number of commits. This could be because of the low number of commits; the project becomes complex with the increased number of commits. With the increase in the commit log for each project an increase in the variety of words is observed.

Table 12: Number of Commits - NLTK

Sentiment	Large	Average	Low
Negative	14.80%	15.56%	9.27%
Neutral	59.53%	54.81%	76.85%
Positive	25.67%	29.63%	13.88%

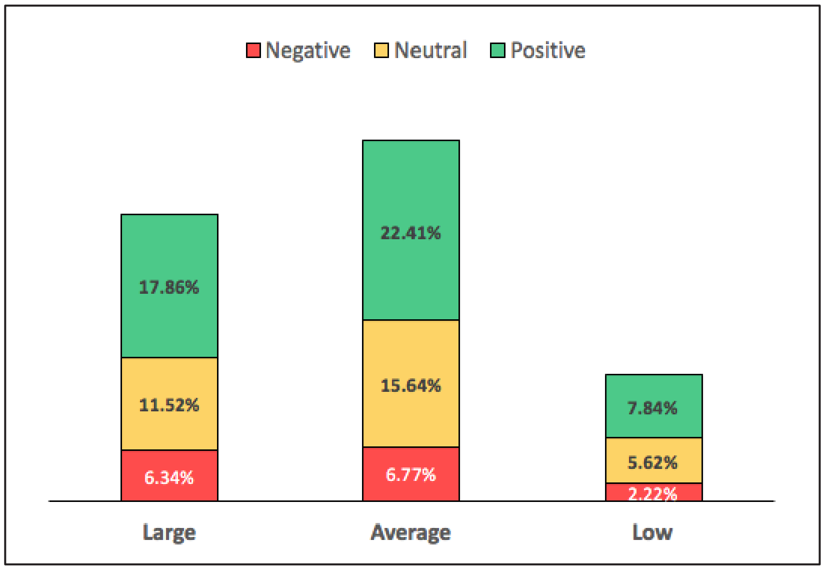


Figure 28: Sentiment Variance for Large, Average, and Low Data sets

We also compared the sentiment polarity from both tools with respect to the day of the week. For RQ 2 we determined maximum positive and maximum negative sentiments. But in order to compare the two tools it was required to categorize the SentiStrength score as positive, negative and neutral sentiments. Figure 29 shows the distribution of sentiments in commit logs with respect to the day of the week using SentiStrength. Figure 30 shows the distribution of sentiments in commit logs with respect to the day of the week using NLTK. It was observed that the positive sentiments on Monday are greater than negative sentiments on using SentiStrength to calculate sentiments where are the NLTK gives just the opposite result. The variance between the positive sentiment for the two tools was less than 1%. NLTK displays maximum negative sentiments on Tuesday where as SentiStrength shows maximum negative sentiments on Wednesday.

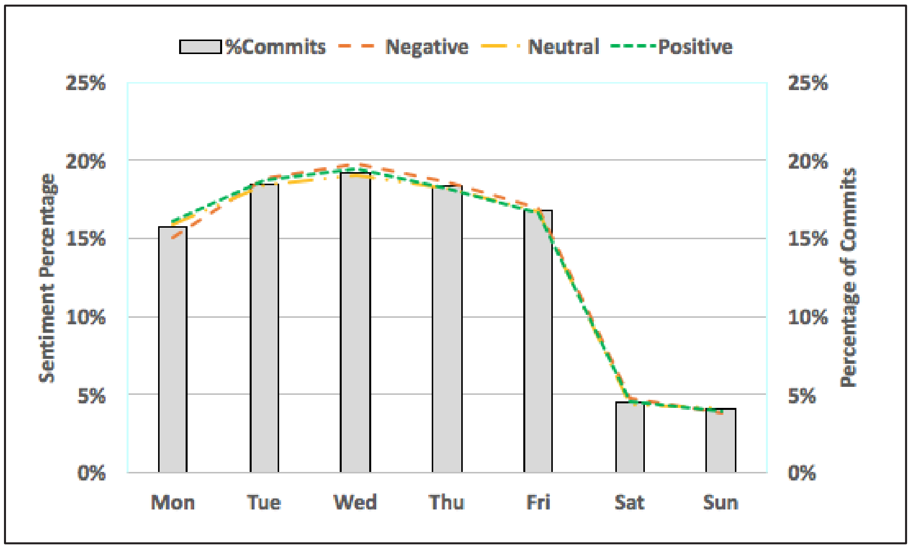


Figure 29: Positive, negative and neutral sentiments across all projects with respect to the day of the week – SentiStrength

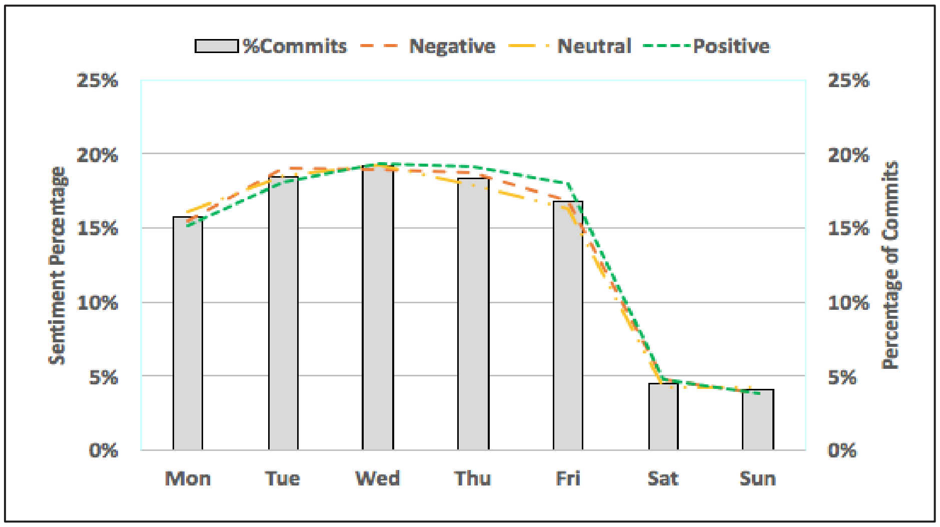


Figure 30: Positive, negative and neutral sentiments across all projects with respect to the day of the week – NLTK

Figure 31, Figure 32 and Figure 33 shows a further breakdown comparison of SentiStrength and NLTK across the three categories for negative sentiments. For projects having maximum number of commits have a very similar trend of sentiments throughout the week for both NLTK and SentiStrength. It was also observed for projects having average number of commit logs that NLTK have low negative sentiments percentage on Wednesday when compared with SentiStrength. This could be because each tool might have a different sentiment polarity for the words being processed in them. Overall, we can see a similar trend as when we compare negative sentiments for both the tools. Jongeling et al. [4] also evaluated based on their research that both SentiStrength and NLTK show consistent results and also have a high agreement with each other.

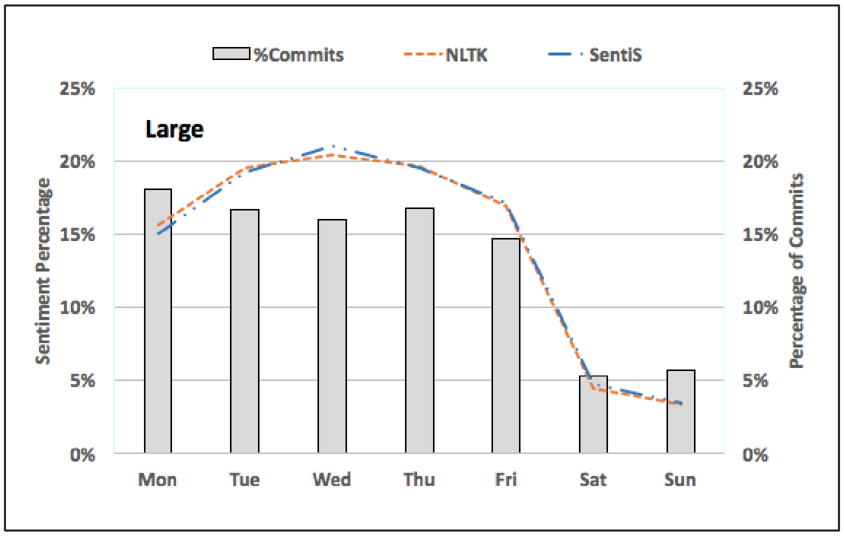


Figure 31: Negative sentiments for projects having maximum number of commits with respect to the day of the week

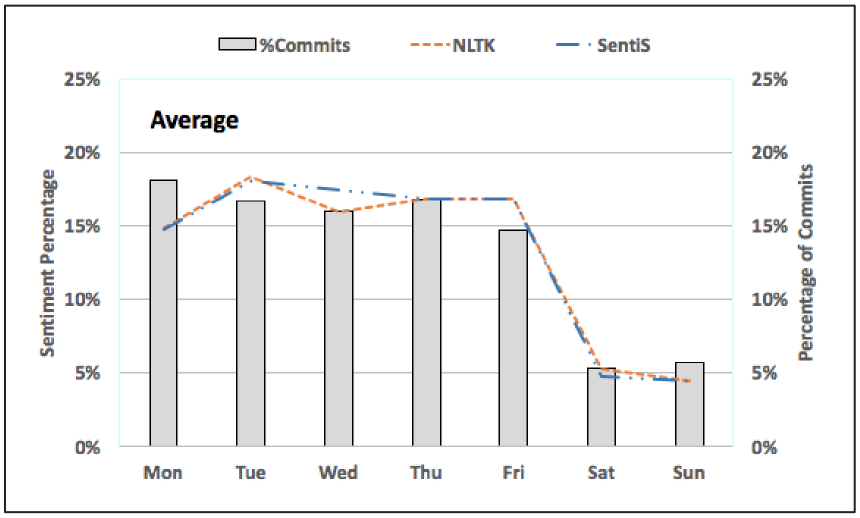


Figure 32: Negative sentiments for projects having average number of commits with respect to the day of the week

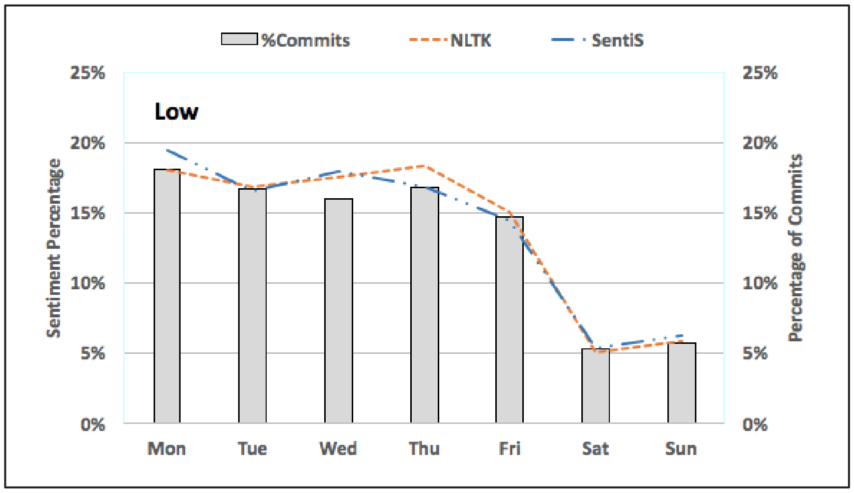


Figure 33: Negative sentiments for projects having low number of commits with respect to the day of the week

We also compared results from RQ 3 for both SentiStrength. Figure 34 shows Positive, negative and neutral sentiments across all projects with respect to average changed files per commit. We observed a spike in the positive sentiments for the year

2010. When we compared the result from NLTK (Figure 34) with SentiStrength (Figure 24) it was observed that there is an increase in positive sentiments with respect to number of changed files per commit (starting from the year 2008 to 2014) while the negative sentiments follow a similar trend. We do notice that NLTK the maximum positive sentiments were reported in the year 2010 with respect to number of changed files per commit; this followed a similar trend with the results obtained by SentiStrength. SentiStrength gives more percentage of negative sentiments than positive sentiments while NLTK gives more percentage of positive sentiments than negative sentiments for the year 2010. This could be because of 15.21% increase in the positive sentiments as shown in Table 11.

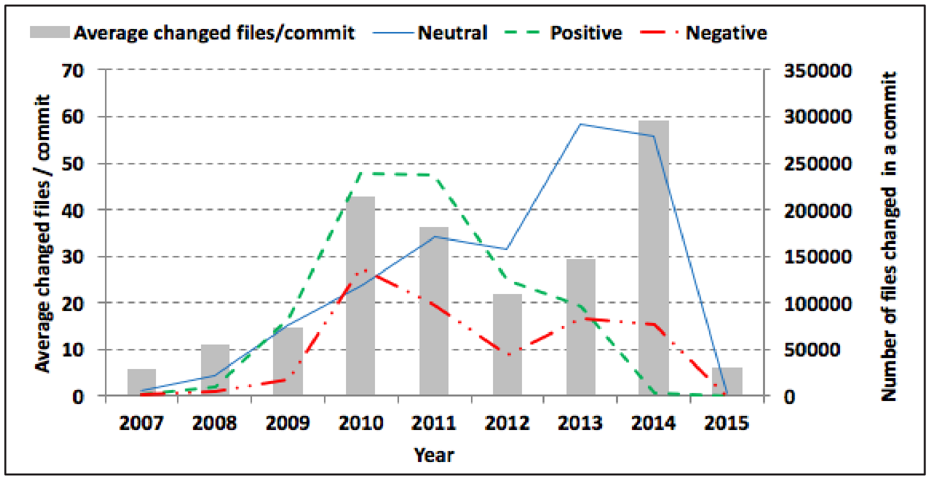


Figure 34: Positive, negative and neutral sentiments across all projects with respect to average changed files per commit - NLTK

4.5 Discussion of Results

As with any real world data, we found the commit logs to contain some questionable values. We found dates that stem from 1970 as well as dates that were in the future such as 2025. These were removed during the analysis of RQ2 and RQ3. Comparing our results of RQ1 to tweets [34] we find that our GitHub commits have 18% (from Table 8) negative sentiment, tweets about scientific papers and tweets about agile project management tools had 1% and 11% negative sentiment respectively. The positive sentiment is 7.199%, 4.20%, and 42% for our analysis of GitHub commits, tweets on scientific papers, and tweets about agile project management tools respectively. Clearly, a lot more negative sentiment is expressed in GitHub commit logs when compared to twitter logs.

Guzman et al. [1] also look into the sentiment of developers by day of week (RQ2). They report that commit logs submitted on Monday have a more negative emotion than commits submitted on any other working day of the week. We conclude that Tuesday was the most negative day overall for all commits. Since we used different datasets than [1], we can't necessarily compare these results directly. Our results for RQ3 provide strong correlation between the number of files changed and the sentiment carried by the commit that contained the files.

However, more work is needed in this area to clearly understand how this relationship impacts the project as a whole. We believe our results are a first step in this direction. We also found more negative sentiment in prior years than more recent years which could be indicative of project stability. Our results for RQ4 that compares the observations from both the tools (NLTK and SentiStrength) demonstrates that the trends are consistent. This clearly indicates that there is a high degree of agreement between the NLTK and SentiStrength. Jongeling et al. [4] analyzed four different sentiment tools to understand to what extent do different sentiment analysis tools agree with emotions of software developers. There was an increase in positive sentiments across all projects with respect to number of changed files per commit (starting from the year 2008 to 2014). It was evaluated and observed that NLTK and SentiStrength give more consistent results compared to other tools [4].

Table 13: Change of Sentiments from SentiStrength to NLTK

SentiStrength	NLTK	% Change
Negative	Negative	7.97%
	Neutral	7.02%
	Positive	3.06%
Neutral	Negative	4.25%
	Neutral	57.02%
	Positive	13.47%
Positive	Negative	0.26%
	Neutral	1.06%
	Positive	5.88%

Table 13 shows a further breakdown of the sentiments for all the commit logs after processing it through NLTK as discussed in Table 11 and allows us compare the change of sentiments (Negative, Neutral and Positive) from SentiStrength to NLTK. It was observed that both the tools show high degree of aggrement as the maximum percentage change can be seen when the sentiments are same in all the three categories. It was also observed in Table 11 that positive sentiments have the maximum variance. 13.47% of neutral commit logs from SentiStrength were categorized as positive when processed through NLTK. In order to understand the variance from neutral to positive we further looked at the commit log for this case.

```
"creating new snapshot release"
```

The above mentioned commit log is taken from project ID `“13803914”` and revision ID `“bb9c24d59c96a2bbf08fa98acc204ccdfe9e44e5”`. When this commit log was processed through NLTK, it determines the log has the positive sentiment whereas when the same commit log processed through SentiStrength, tells us that the commit log is neutral. Considering the two tools use different algorithms to calculate the sentiment polarity of the text we cannot determine at this point which one is better. We present another example where the overall sentiment of the commit log when processed with SentiStrength is neutral but when processed by NLTK is positive.

```
“improving a bit warble default config comments”
```

The above mentioned commit log is taken from project ID `“2019”` and revision ID `“20ebd9e49d34ecbfe26243a95471c91eedb72325”`. Comparing both the above mentioned commit logs we can assume that NLTK focuses on each word of the text and determines the sentiment. In the second example the word “improving” determines a positive emotion similarly the word “creating” bends the sentiment towards the positive emotion while SentiStrength calculates the score of each word and finds the maximum positive score and maximum negative score. In addition, SentiStrength might have a better algorithm to understand the sentiment of the complete text processed through it.

One way to determine which tool works the best, would be to conduct a human-based study and ask developers what sentiment they think is portrayed by the commit message. This sentiment as evaluated by human developers can then be compared to the sentiment a tool outputs. Such a study is planned as future work. Doing such a comparison will also help in improving sentiment analysis tools by fine tuning them to software engineering artifacts where the vocabulary is different from other tradional artifacts such as tweets and blog posts where sentiment analysis is most often used.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The thesis presents a study of sentiment analysis on GitHub commit logs. We found that a majority of the sentiment in GitHub projects are categorized as neutral but when comparing positive with negative sentiment, we found more than twice the percentage of negative sentiment than positive ones when analyzing all the commit logs in the specified dataset. Overall, more negative sentiment was detected on Tuesday, however, for the top five projects with the most commits, Wednesdays and Thursdays were the most negative. There is positive correlation with sentiment and the number of files changed. Finally, we found consistent results on comparing NLTK and SentiStrength. Future work can look into the specific type of file change (such as an addition, deletion or modification of a file) to determine if any relationship exists. In the future, we plan on replicating this study using the GitHub large dataset and other sentiment analysis tools. We will also work towards training sentiment tools on a validated set of commit messages to make them more robust for software engineering problems. In addition, we also plan to collect few commit logs based upon the SentiStrength score, NLTK probability and the overall commit log emotion and conduct a survey with a group of people thereby providing us with some insight into the quality of the tools as perceived by human evaluators. This will enable us to compare the emotions that humans perceive with how sentiment analysis tools such as SentiStrength and NLTK thereby helping us understand the accuracy for each tool.

References

[1] E. Guzman, D. Azócar, and Y. Li, “Sentiment analysis of commit comments in GitHub: an empirical study,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 352–355.

[2] P. Tourani, Y. Jiang, and B. Adams, “Monitoring sentiment in open source mailing lists-exploratory study on the apache ecosystem,” in *Proceedings of the 2014 Conference of the Center for Advanced Studies on Collaborative Research (CASCON), Toronto, ON, Canada*, 2014, pp. 74–95.

[3] A. Murgia, P. Tourani, B. Adams, and M. Ortu, “Do developers feel emotions? an exploratory analysis of emotions in software artifacts,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 262–271.

[4] R. Jongeling, S. Datta, and A. Serebrenik, “Choosing your weapons: On sentiment analysis tools for software engineering research,” in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, 2015, pp. 531–535.

[5] V. Sinha, A. Lazar, and B. Sharif, “Analyzing Developer Sentiment in Commit Logs,” in *The 13th International Conference on Mining Software Repositories (MSR 2016)*, 2016.

[6] “Dictionary.com | Find the Meanings and Definitions of Words at Dictionary.com.” [Online]. Available: http://www.dictionary.com/. [Accessed: 19-Apr-2016].

[7] A. Sarlan, C. Nadam, and S. Basri, “Twitter sentiment analysis,” in *2014 International Conference on Information Technology and Multimedia (ICIMU)*, 2014, pp. 212–216.

[8] E. F. Andrew N. Smith, “How Does Brand-related User-generated Content Differ across YouTube, Facebook, and Twitter?,” *J. Interact. Mark.*, vol. 26, no. 2, pp. 102–113, 2012.

[9] “A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle.” [Online]. Available: http://clair.eecs.umich.edu/aan/paper.php?paper_id=P12-3020. [Accessed: 22-Mar-2016].

[10] A. Abbasi, A. Hassan, and M. Dhar, “Benchmarking Twitter Sentiment Analysis Tools.,” in *LREC*, 2014, pp. 823–829.

[11] R. Pfitzner, A. Garas, and F. Schweitzer, “Emotional Divergence Influences Information Spreading in Twitter.,” *ICWSM*, vol. 12, pp. 2–5, 2012.

[12] E. Kouloumpis, T. Wilson, and J. Moore, “Twitter Sentiment Analysis: The Good the Bad and the OMG!,” presented at the AAAI Press, 2011.

[13] S. Hangal, M. S. Lam, and J. Heer, “MUSE: Reviving Memories Using Email Archives,” in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 2011, pp. 75–84.

[14] P. Matykiewicz, W. Duch, and J. Pestian, “Clustering Semantic Spaces of Suicide Notes and Newsgroups Articles,” in *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, Stroudsburg, PA, USA, 2009, pp. 179–184.

[15] S. M. Mohammad and T. (Wenda) Yang, “Tracking Sentiment in Mail: How Genders Differ on Emotional Axes,” in *Proceedings of the 2Nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, Stroudsburg, PA, USA, 2011, pp. 70–79.

[16] C. O. Alm, D. Roth, and R. Sproat, “Emotions from Text: Machine Learning for Text-based Emotion Prediction,” in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA, 2005, pp. 579–586.

[17] N. Cheng, X. Chen, R. Chandramouli, and K. P. Subbalakshmi, “Gender identification from E-mails,” in *IEEE Symposium on Computational Intelligence and Data Mining, 2009. CIDM ’09*, 2009, pp. 154–158.

[18] M. Corney, O. de Vel, A. Anderson, and G. Mohay, “Gender-preferential text mining of e-mail discourse,” in *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, 2002, pp. 282–289.

[19] B. Boneva, R. Kraut, and D. Frohlich, “Using E-mail for Personal Relationships The Difference Gender Makes,” *Am. Behav. Sci.*, vol. 45, no. 3, pp. 530–549, Nov. 2001.

[20] “The consumer’s reaction to delays in service,” *Int. J. Serv. Ind. Manag.*, vol. 13, no. 2, pp. 118–140, May 2002.

[21] S. C. Müller and T. Fritz, “Stuck and Frustrated or in Flow and Happy: Sensing Developers’ Emotions and Progress,” in *Proceedings of the 37th International*

Conference on Software Engineering - Volume 1, Piscataway, NJ, USA, 2015, pp. 688–699.

[22] E. Guzman and B. Bruegge, “Towards Emotional Awareness in Software Development Teams,” in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, New York, NY, USA, 2013, pp. 671–674.

[23] G. Gousios, “The GHTorent Dataset and Tool Suite,” in *Proceedings of the 10th Working Conference on Mining Software Repositories*, Piscataway, NJ, USA, 2013, pp. 233–236.

[24] M. Thelwall, K. Buckley, and G. Paltoglou, “Sentiment Strength Detection for the Social Web,” *J Am Soc Inf Sci Technol*, vol. 63, no. 1, pp. 163–173, Jan. 2012.

[25] M. De Choudhury and S. Counts, “Understanding Affect in the Workplace via Social Media,” in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, New York, NY, USA, 2013, pp. 303–316.

[26] “Facilitating software evolution research with kenyon.” [Online]. Available: http://dl.acm.org/citation.cfm?id=1081736. [Accessed: 11-Apr-2016].

[27] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: A Distributed Storage System for Structured Data,” *ACM Trans Comput Syst*, vol. 26, no. 2, pp. 4:1–4:26, Jun. 2008.

[28] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, “Boa: A language and infrastructure for analyzing ultra-large-scale software repositories,” in *Proceedings of the 2013 International Conference on Software Engineering*, 2013, pp. 422–431.

[29] “The Boa Programming Guide - Domain-Specific Types - Boa - Iowa State University.” [Online]. Available: http://boa.cs.iastate.edu/docs/dsl-types.php. [Accessed: 11-Apr-2016].

[30] “SAS Product Documentation.” [Online]. Available: http://support.sas.com/documentation/. [Accessed: 19-Apr-2016].

[31] D. Pletea, B. Vasilescu, and A. Serebrenik, “Security and Emotion: Sentiment Analysis of Security Discussions on GitHub,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, New York, NY, USA, 2014, pp. 348–351.

[32] “vaderSentiment 0.5 : Python Package Index.” [Online]. Available: https://pypi.python.org/pypi/vaderSentiment. [Accessed: 13-Apr-2016].

[33] “Sentiment Analysis using Vader,” *LinkedIn Pulse*, 01-Jan-2016. [Online]. Available: https://www.linkedin.com/pulse/sentiment-analysis-using-vader-muthuraj-kumaresan. [Accessed: 13-Apr-2016].

[34] N. Friedrich, T. D. Bowman, W. G. Stock, and S. Haustein, “Adapting sentiment analysis for tweets linking to scientific papers,” *ArXiv Prepr. ArXiv150701967*, 2015.