**COMPUTER PROJECT 2**

**Project 3**

**Group 11**

Emre Can Bulut - 150130705

Önder Postoğlu - 150120013

Hasan Hüseyin Çağlar - 150110042

**INTRODUCTION**

In this project, we were asked to mine data on Github. We extracted commit-based data of Chromium project on Github, formatted the raw data in a way that we could use. The project itself is almost 10 gigabytes and even the commit history information is measured with megabytes. To be able to handle the situation, we narrowed down the period of when the commits are made so that the log file would be in an affordable size. That, in our case, was a period of 15 days. The detailed explanation of mining and usage of data will be given in the following sections.

**Mining Chromium’s Version Control System**

Before the project, we had no idea about how to mine a repository. So, after some research, we came up with 2 options. We would either use *HTML Parser* or *git log*. We decided on sticking to the second option. And, we cloned the repository to our computers via below command:  
*git clone* [*https://chromium.googlesource.com/chromium/src*](https://chromium.googlesource.com/chromium/src)Then, the git log command could be used. However, we needed to decide on which data we would need and in which format we would need it. After deciding on using “author” and “changed files”, we made sure that every commit was unique by adding “commit id” as well. And the resulting git log command is as following:

*function getcommit { \*

*git show --pretty="format:" --name-only $1 | \*

*perl -pe's/^\n//g;' | \*

*sed 's/\(.\*\)/"\1"/g' | \*

*perl -0pe 's/\n(?!\Z)/,\n/g'; \*

*}*

*export -f getcommit*

*git log --after="2016-04-25" --pretty=format:'{%n "commit": "%H",%n "author": "%an",%n "files": [ COMMIT\_HASH\_%H ]%n},' | \*

*perl -pe 'BEGIN{print "["}; END{print "]\n"}' | \*

*perl -pe 's/},]/}]/;s/COMMIT\_HASH\_(\w+)/`echo -n "";getcommit $1`/e' -> generated.json*

Using this command, a JSON file was generated which made it very easy to go through the data.

**Identifying Top Developers**

To identify the top developers, we counted how many commits there were in total in that period. Then, the commit rate was calculated individually for each developer. After extracting the data to an excel file, we plotted the graph of authors to their commit numbers. (Codes for extracting the data to an excel file and the excel file are submitted)

**Identifying Edited File Sets in Commits and Forming Adjacency Matrix**

Edited file sets in commits were already written in the JSON file that was extracted. So, all we needed to do was to go through the file, and create a matrix. In this matrix, each row represented a file and each column represented a developer as specified in project description. Once the matrix was formed, we continued with adjacency matrix using the information in the previous matrix.

Creating adjacency matrix had a higher cost in terms of complexity. Therefore, the symmetry property of adjacency matrix was used. We filled the lower triangular of the adjacency matrix with the help of the previous matrix. Then, the upper triangular half was filled by the values from lower triangular half of the matrix.

**Visualization of Socio-technical Network**

This was the most diffucult part since we were dependent on Visualization applications. Every visualization app. wants the input matrix in a specific format. So, after going through several applications, SocNetV worked without any problem. And, using the GUI, we edited and filtered which nodes and edges to see since the edges were covering the entire board. After the network was created, it was very easy to see who colloborated with others the most and who has very little to no colloboration.

**Links**

Youtube Presentation - <https://www.youtube.com/watch?v=1hlPXh4RvfE>

Github Repo - <https://github.com/emrecanbulut/ComputerProject2>