



MINING CHROME REPOSITORY

Project Report
May 2016

Project Summary

Project Duration:
4 weeks

Project Workload:
40 hour/man

Team Workers in Group 16:
040110078 Nurefşan Sertbaş
040090508 Betül Kantepe

Project Leader:
*Team leader did not work on the
project

Project Github Link:

https://github.com/sertbasn1/DM_Project3.git

Project Youtube Link:

<https://youtu.be/OdUMpZixfPE>

CONTENTS

1.Project Description	5
2.Introduction to Data Mining	5
2.1. About Chromium Project	5
2.2. About Version Control Systems	5
2.3. About Writing Scripts by Using Git Commands	6
3.The Process of Data Mining.....	6
3.1. Identifying Top Developers.....	9
3.1.1. Plot the distribution of commits in terms of each developer	9
3.1.2. Commit frequency of developers	20
3.1.3. Identifying top developers who make %80 of commits.....	25
3.2. Identifying Edited File Sets in Commits	27
3.2.1. Creating matrix	27
3.3. Building and Visualization of Socio-technical Network of Chrome Project	30
4.Sample Execution Result	32
5.References	33

List of Figures

- Figure 1: Extracting and saving all authors of the project
- Figure 2: The capture of allauthors.log file including all author names
- Figure 3: Extracting all commits and commit dates for each developer
- Figure 4: The capture of commitdateandauthor.log file including commit dates
- Figure 5: Extracting the number of commits in a specific time interval
- Figure 6: The capture of commitof2months.log file including commit nums of 2 months
- Figure 7: The function call from C
- Figure 8: The capture of commitof2months_c1.txt file including commit percentage
- Figure 9: The capture of commitof2months_c2.txt file including commit numbers
- Figure 10: Finding number of commits by using Git command
- Figure 11: The capture of commitcountandauthors1.log file including commit counts
- Figure 12: The function call for calculating commit frequency for each developer
- Figure 13: The capture of commit frequencies in commitfrequencyofdevelopers.txt
- Figure 14: Using Git command for finding top developers
- Figure 15: The capture of commitcountandauthors.log including descending order
- Figure 16: The function call to calculate top developers
- Figure 17: The capture of topdevelopers.txt including developers with commit nums
- Figure 18: The code to generate the files/developers matrix
- Figure 19: The capture of allchangedfilesandauthors.log file
- Figure 20: The Git command to list all the files in the repo and save into a file
- Figure 21: The capture of allfiles.log including all files in the repo
- Figure 22: The C++ function call to preprocess the matrix
- Figure 23: The C++ function call to create the matrix
- Figure 24: The capture of the matrix including files/developers info
- Figure 25: The social network analysis graph for developers done by UCINET tool
- Figure 26: The sample execution result for the project
- Figure 27: The window of “Convert two mode to one mode data” in UCINET 6

List of Tables

Tables 1-19: The distribution of commit numbers for all developers

Tables 20-25: The distribution of commit numbers for all developers

Tables 26-28: The plot of top developers with commit numbers

1. Project Description

Project aims to identify social and technical dependencies in Chrome project. Students are expected to extract commit-based information from version control systems of Chrome project, such as edited file sets, developers who made the commit, and co-changed files in the commits. Based on those information, a visual representation of technical dependencies and social dependencies will be done.

2. Introduction to Data Mining

Data mining is the process which provides retrieving of hidden useful information from large databases. Furthermore, it is commonly used by companies in order to determine relationships between factors such as price, product, etc.

In this project, we used data mining approach so as to mine Chromium's version control system by writing scripts. For this aim, we followed below steps:

- Identify top developers
- Identify edited file sets
- Build socio-technical network of Chrome project

2.1 About Chromium Project

In this project, we analyzed the Chromium project in detail. We mined 5984 commits between dates 01-01-2010 and 01-03-2010 which means two months.

We decided to analyze the project by specifying time interval instead of choosing specific directory since it is more logical to take into account whole development of the project. With this approach, the outputs which we have obtained by mining of the project provides us a general perspective about project development history.

Since the chromium project has been developing by using Git version control system, we extract the commits of the project thanks to Git. Before the extraction process of commits, we have checked where and how the project is located in Github.

2.2 About Version Control Systems

Version control systems are useful tools by which development teams can manage the changes made to the source code beside owner of the change and references to problems fixed.

One of the commonly used VCS tools today is called Git. Similar to other popular VCS systems, Git is free and open source. We have used Git in order to track the history of development in specified time interval.

2.3 About Writing Scripts by Using Git Commands

There are several ways to use Git commands such as graphical user interface and command line. In this project, we preferred to use Git from command line so we wrote our scripts by using Bash scripting language with using directly outputs of the Git commands. Also, we benefit from C and C++ function calls in the script with the aim of providing convenience.

In order to achieve the processes which are given in the project description, we followed the below steps respectively.

Firstly, we used Git commands directly in bash script then we saved the outputs into text files. After, we processed these files by using code snippets written in C and C++.

3.The Process of Data Mining

As we mentioned above, we have chosen specific development period as 01-01-2010 to 01-03-2010 and extracted raw data which includes information about committer, commit date, content of the commit and manipulated files in that commit. Then, we started to process the extracted data to obtain useful information.

For this aim, we wrote below Bash scripts as a preprocess.

First of all, we extract all the authors which are actively committed in chosen time range with following commands and we saved the results into `allauthors.log` file under the `outputs` directory.

Note that, we collect all desired outputs under the *outputs* directory.

```
echo Authors are extracting..
git log --format='%aN' --since=2010-01-01 --before=2010-03-01 | sort -u >outputs/allauthors.log

getauthors() {
authors=() # Create array
while IFS= read -r line # Read a line
do
authors+=("$line") # Append line to the array
done < "$1"
}

getauthors "outputs/allauthors.log"
for e in "${authors[@]}"
do
git shortlog --since=2010-01-01 --before=2010-03-01 --author="$e" >>outputs/commitandauthors.log
done
```

Figure 1: Extracting and saving all authors of the project

The capture of allauthors.log file is given in below screenshot.

```
senorblanco@chromium.org
sfalken@apple.com
sgjesse@google.com
shess@chromium.org
siggi@chromium.org
simon.fraser@apple.com
skerner@chromium.org
skerner@google.com
skrul@chromium.org
sky@chromium.org
slewis@apple.com
slightlyoff@chromium.org
snej@chromium.org
steveblock@google.com
stoyan@chromium.org
stuartmorgan@chromium.org
sullivan@apple.com
suzhe@chromium.org
thakis@chromium.org
thestig@chromium.org
thomasvl@chromium.org
tim@chromium.org
timothy@apple.com
```

Figure 2: The capture of allauthors.log file including all author names

Secondly, by using allauthors.log file which contains all the names of the authors, we extracted all the commits and commit dates for each committers in the mentioned log file below and we saved the results into commitdateandauthor.log file under the outputs directory.

```
echo Total commits with commit dates are extracting..
for e in "${authors[@]}"
do
echo "$e" >>outputs/commitdateandauthor.log
git log --author="$e" --since=2010-01-01 --before=2010-03-01 | grep Date | awk '{print " : \"$4\" \"$3\" \"$6\"}' |
    uniq -c >>outputs/commitdateandauthor.log
done
```

Figure 3: Extracting all commits and commit dates for each developer

The capture of commitdateandauthor.log file is given in below screenshot.

```

4 : 19 Jan 2010
2 : 12 Jan 2010
1 : 11 Jan 2010
yusukes@chromium.org
1 : 3 Feb 2010
1 : 28 Jan 2010
yusukes@google.com
1 : 22 Jan 2010
1 : 20 Jan 2010
yutak@chromium.org
1 : 28 Jan 2010
yuzo@chromium.org
3 : 23 Feb 2010
1 : 19 Jan 2010
3 : 18 Jan 2010
yuzo@google.com
1 : 22 Feb 2010
2 : 12 Feb 2010
2 : 9 Feb 2010
1 : 4 Feb 2010
zecke@webkit.org
1 : 4 Feb 2010
1 : 28 Jan 2010
zelidrag@chromium.org
1 : 17 Feb 2010
1 : 9 Feb 2010
1 : 5 Feb 2010
1 : 26 Jan 2010
1 : 23 Jan 2010

```

Figure 4: The capture of commitdateandauthor.log file including commit dates

As a last step of the preprocess, we extract how many commits are done in selected time interval by each author and the result is saved into commitof2months.log file under the outputs directory.

```

##Extract data for a specific time interval
echo Extracting data of given time interval..
for e in "${authors[@]}"
do
git shortlog --since=2010-01-01 --before=2010-03-01 --author="$e" >outputs/tmp.log
head -1 outputs/tmp.log >>outputs/commitof2months.log
done

```

Figure 5: Extracting the number of commits in a specific time interval

The capture of commitof2months.log file is given in below screenshot.


```

pkasting@chromium.org (114):
pvalchev@google.com (14):
rafaelw@chromium.org (13):
robert@webkit.org (1):
robertshield@chromium.org (15):
rogerta@chromium.org (2):
rohitrao@chromium.org (25):
rolandsteiner@chromium.org (16):
rsesek@chromium.org (43):
rvargas@google.com (23):
satorux@chromium.org (14):
scherkus@chromium.org (3):
sehr@google.com (14):
senorblanco@chromium.org (25):
sfalken@apple.com (37):
sgjesse@google.com (4):
shess@chromium.org (23):
siggi@chromium.org (3):
simon.fraser@apple.com (52):
skerner@chromium.org (17):
skerner@google.com (1):
skrul@chromium.org (13):
sky@chromium.org (80):
cloud@chromium.org (2):

```

Figure 6: The capture of commitof2months.log file including commit nums of 2 months

3.1 Identifying Top Developers

3.1.1 Plot the distribution of commits in terms of each developer

For this aim, we benefit from code snippet written in C. It takes commitof2months.log file as an input and produces commitof2months_c1.txt and commitof2months_c2.txt. The function call from C is given below.

```

gcc calculate.c -o calculate
./calculate outputs/commitof2months.log

```

Figure 7: The function call from C

The capture of commitof2months_c1.txt file is given in below screenshot. It contains the commit percentage of each author in selected time interval. We used the file when determining the top developers of the project.

```

0.365509:aa@chromium.org
0.016614:abarth@chromium.org
1.279282:abarth@webkit.org
0.232597:abecsi@webkit.org
0.016614:ace@chromium.org
0.083070:adele@apple.com
0.215983:ager@chromium.org
0.681176:agl@chromium.org
0.299053:ajwong@chromium.org
0.481808:akalin@chromium.org
0.182755:albertb@google.com
0.066456:alex@webkit.org
0.016614:alice.liu@apple.com
0.631334:alokp@chromium.org
0.182755:amit@chromium.org
0.897159:ananta@chromium.org
0.033228:andersca@apple.com
0.348895:andybons@chromium.org
0.099684:antonm@chromium.org
0.049842:antonm@google.com
0.016614:antti@apple.com
1.345730:an@apple.com

```

Figure 8: The capture of commitof2months_c1.txt file including commit percentage

The capture of commitof2months_c2.txt file is given in below screenshot. It contains the number of commits of each author in selected time interval. It is important to note that, we will be using this file to visualize the distribution with the help of Matlab.

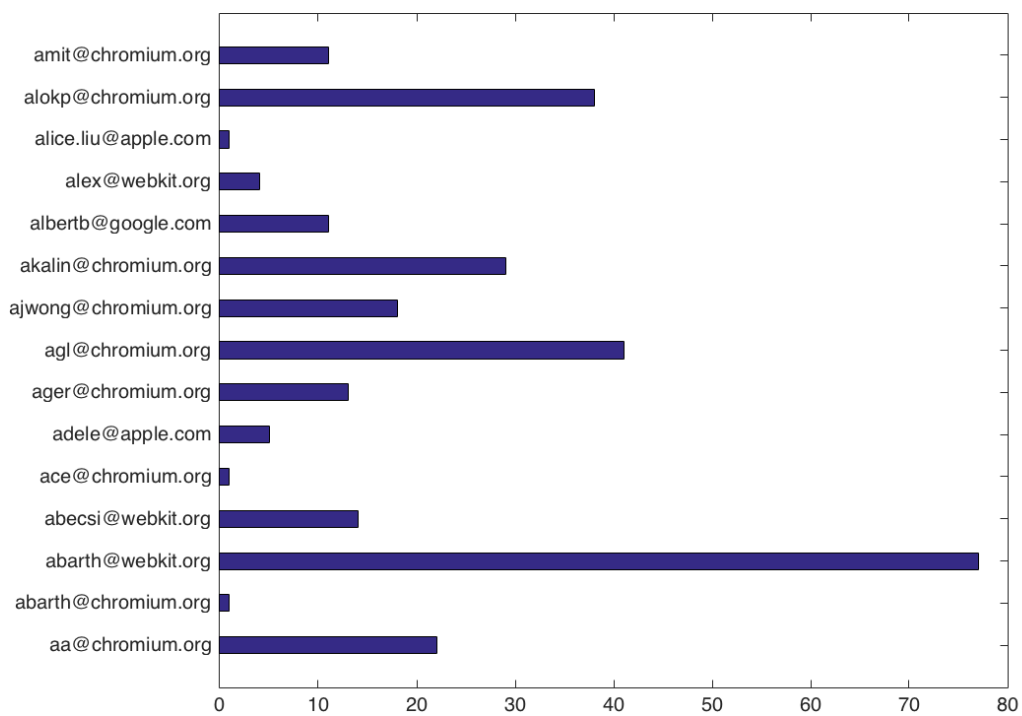
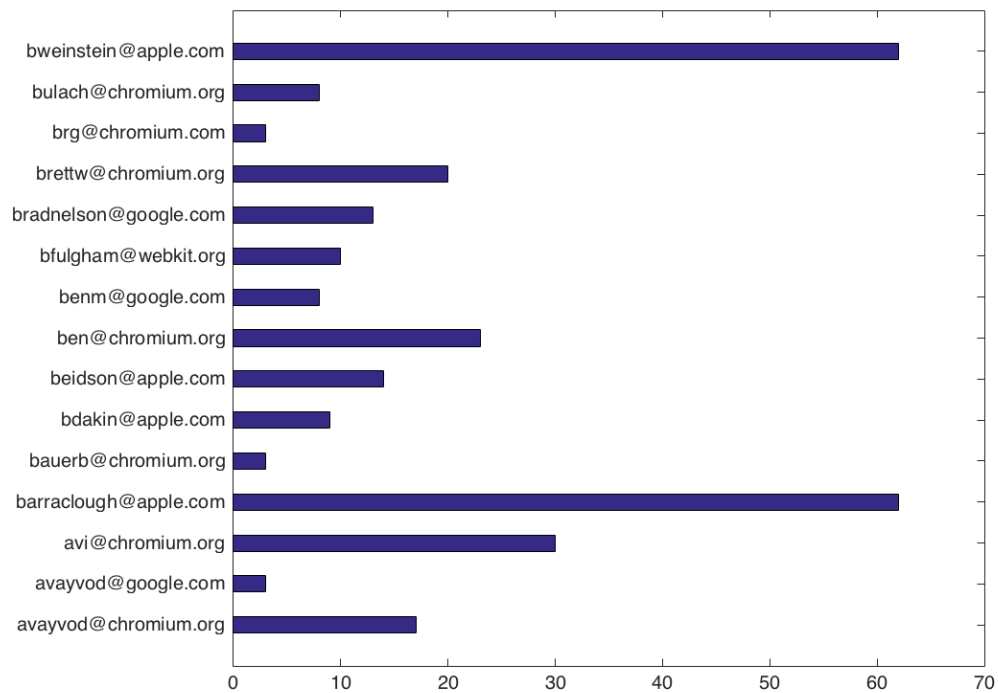
```

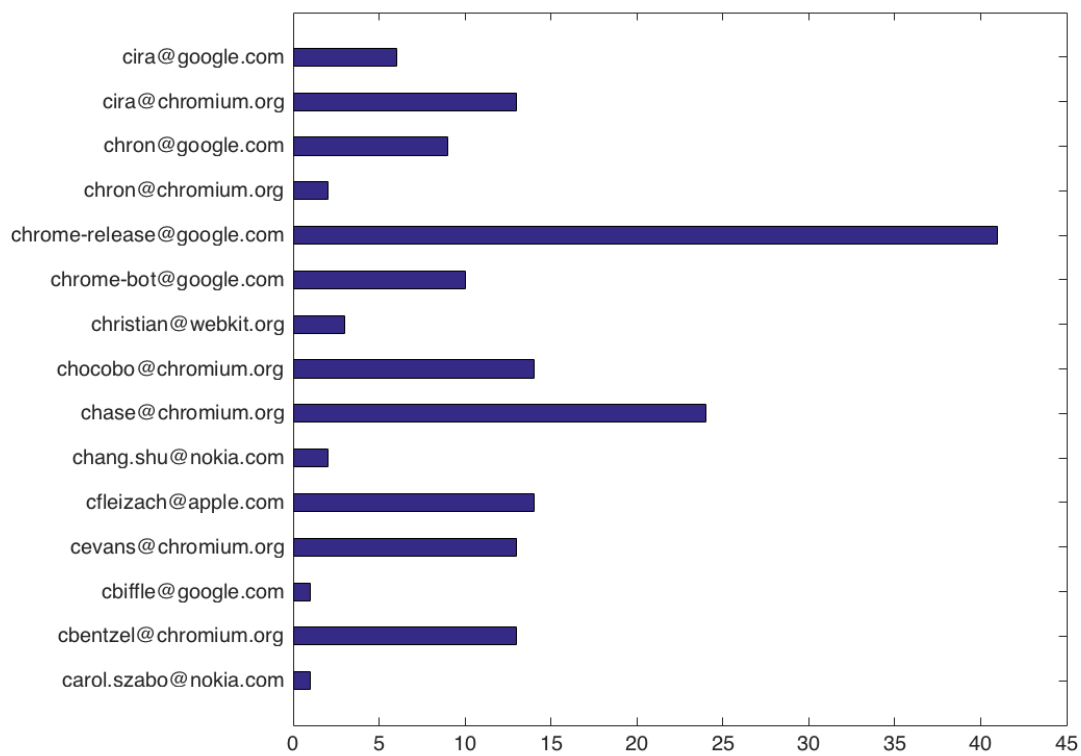
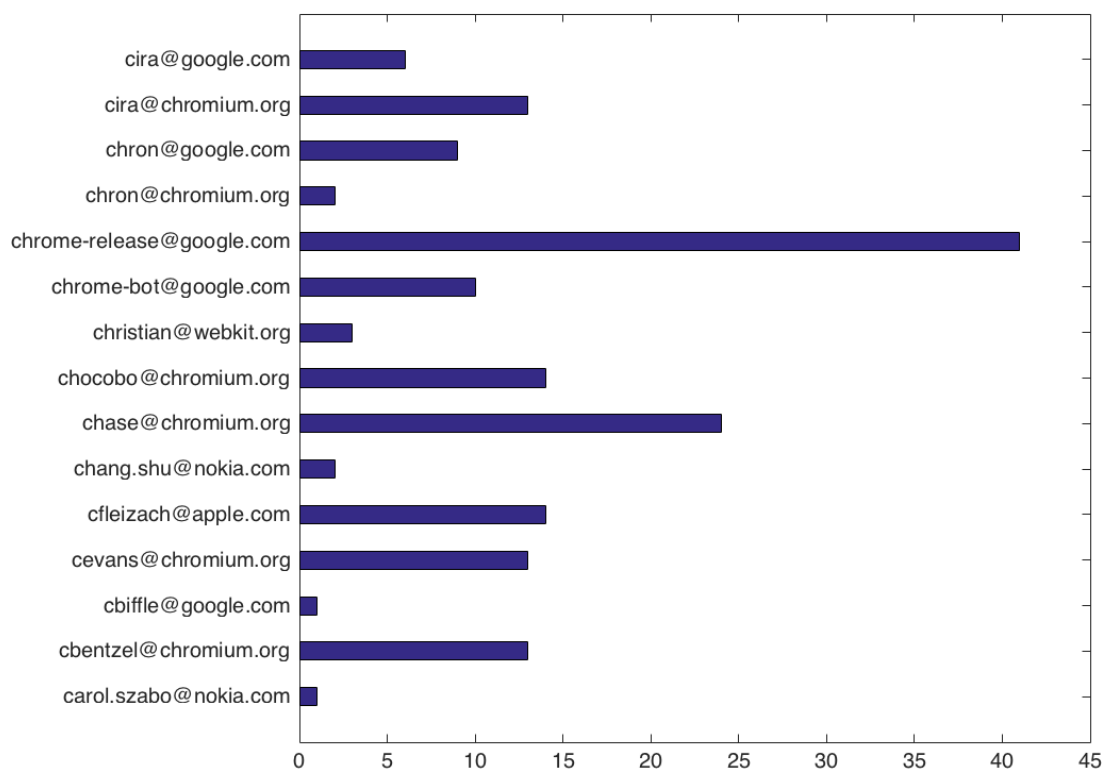
---
22:aa@chromium.org
1:abarth@chromium.org
77:abarth@webkit.org
14:abecsi@webkit.org
1:ace@chromium.org
5:adele@apple.com
13:ager@chromium.org
41:agl@chromium.org
18:ajwong@chromium.org
29:akalin@chromium.org
11:albertb@google.com
4:alex@webkit.org
1:alice.liu@apple.com
38:alokp@chromium.org
11:amit@chromium.org
54:ananta@chromium.org
2:andersca@apple.com
21:andybons@chromium.org
6:antonm@chromium.org
3:antonm@google.com
1:antti@apple.com

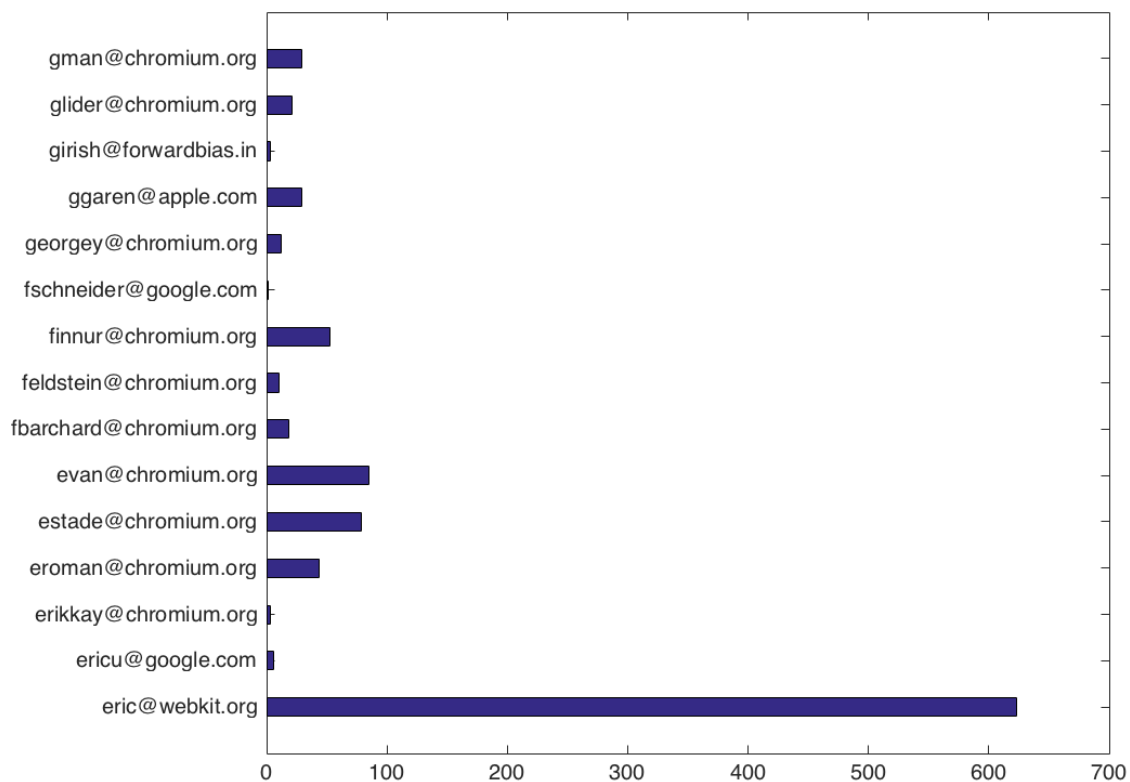
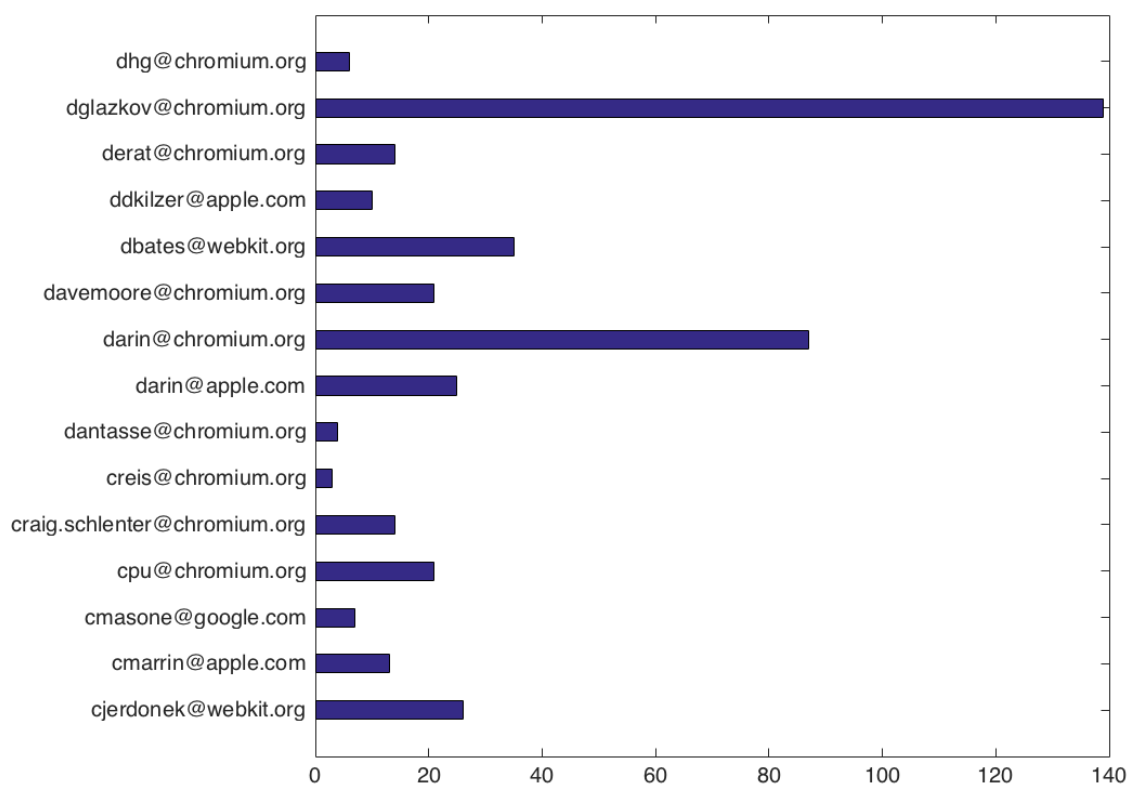
```

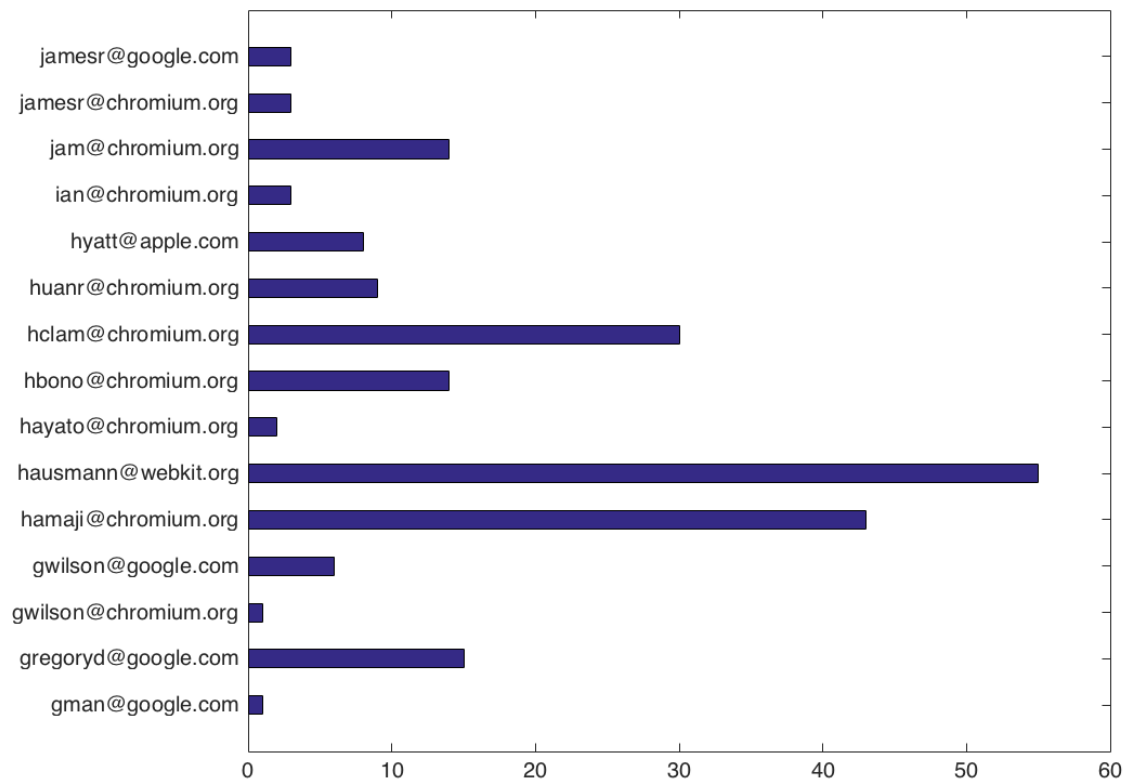
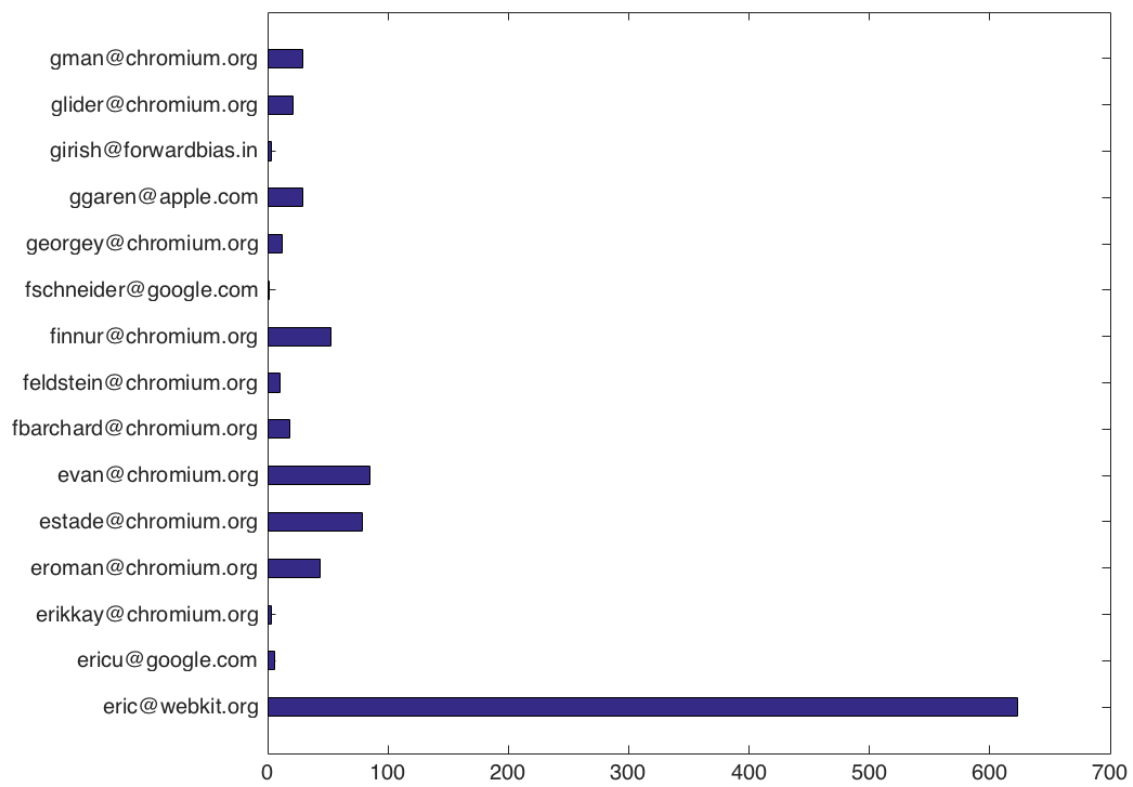
Figure 9: The capture of commitof2months_c2.txt file including commit numbers

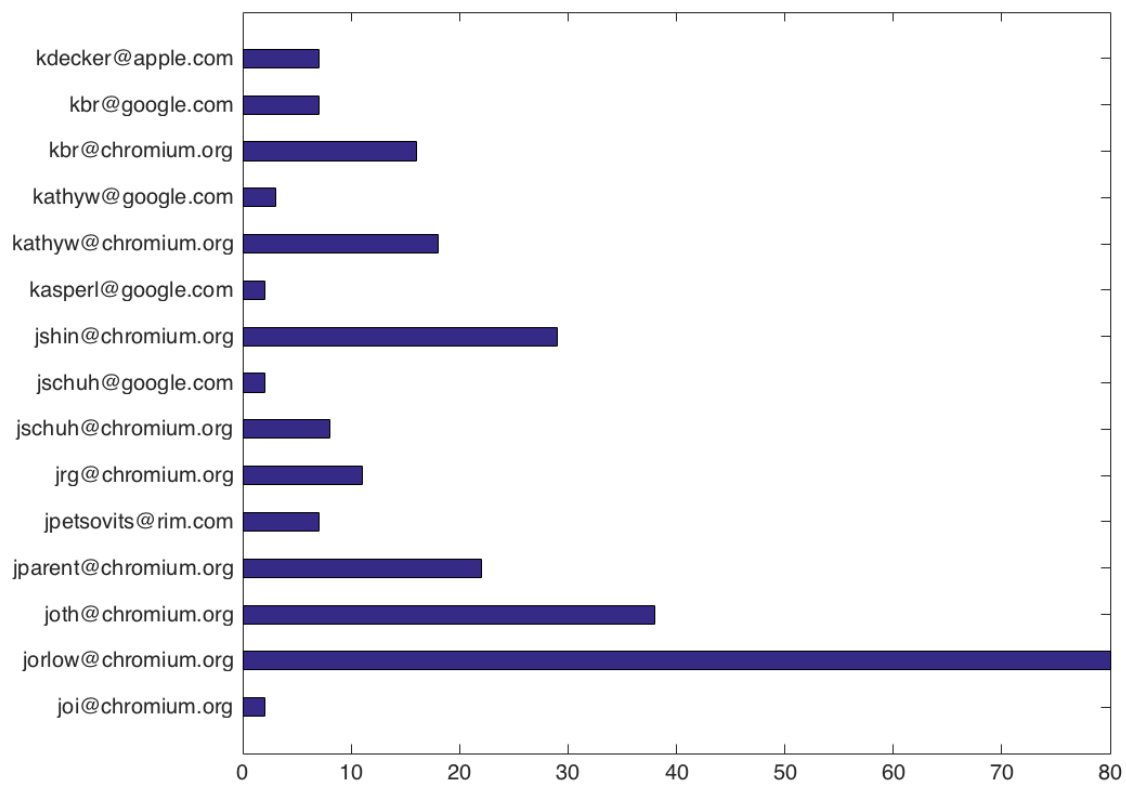
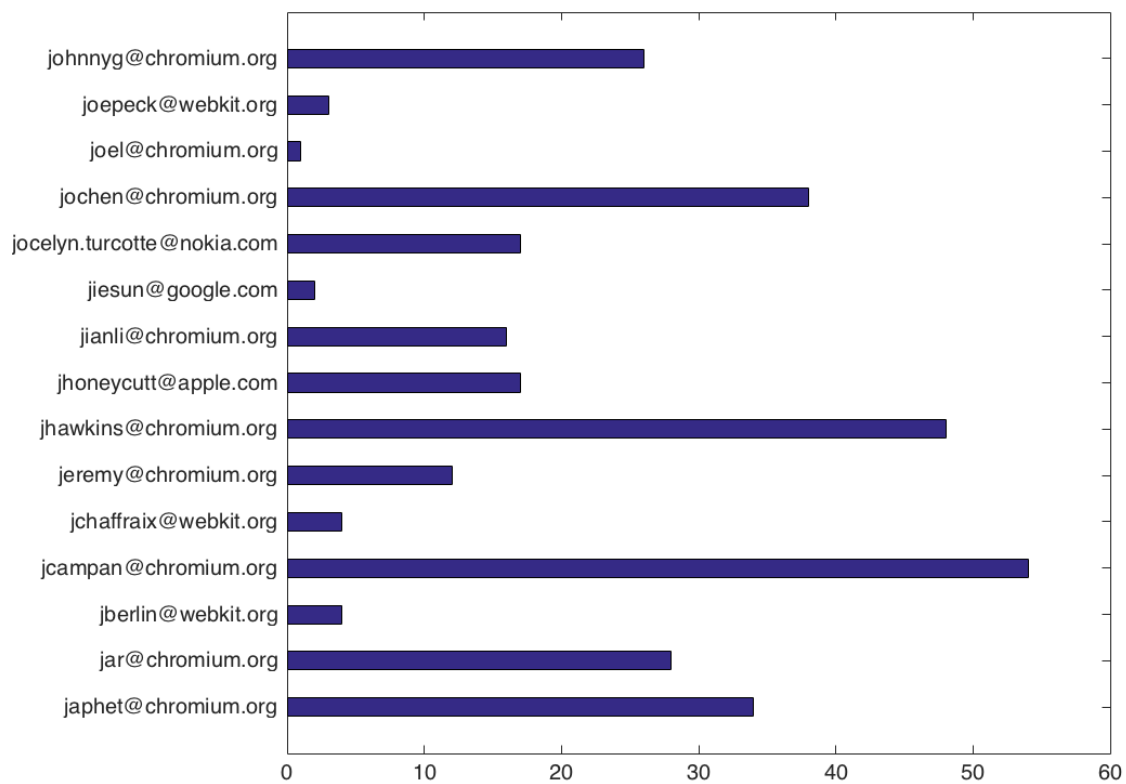
The plot results are given below respectively:

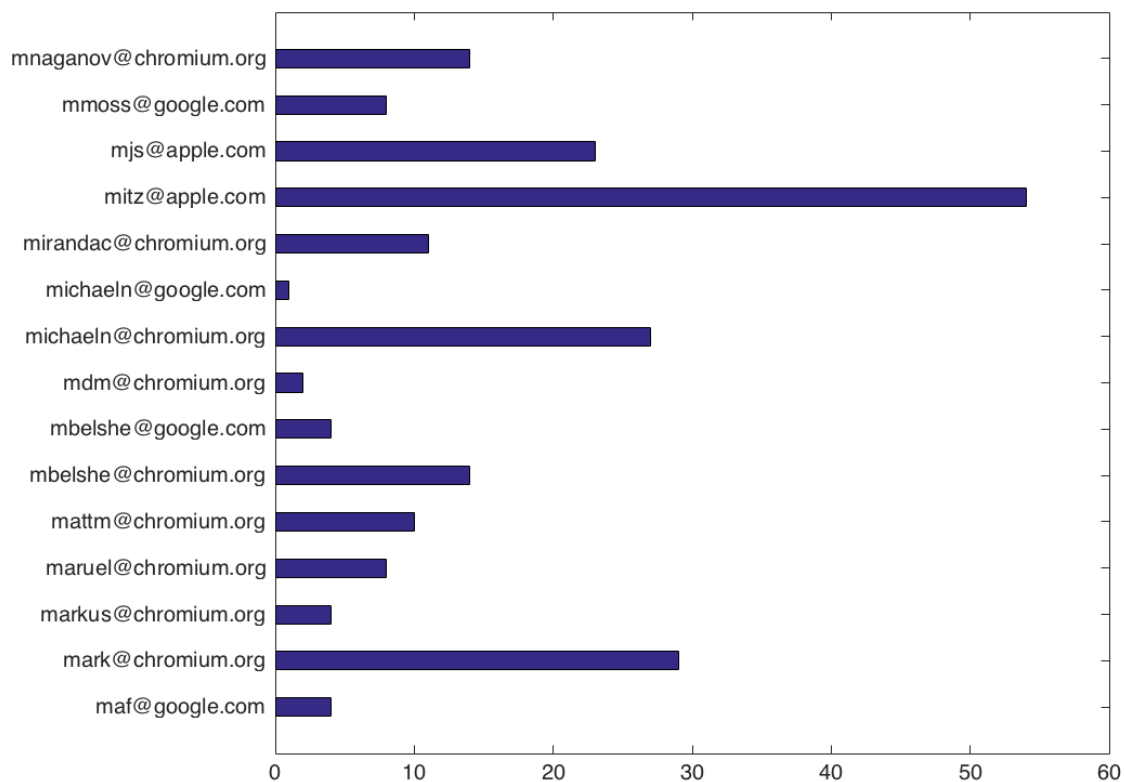
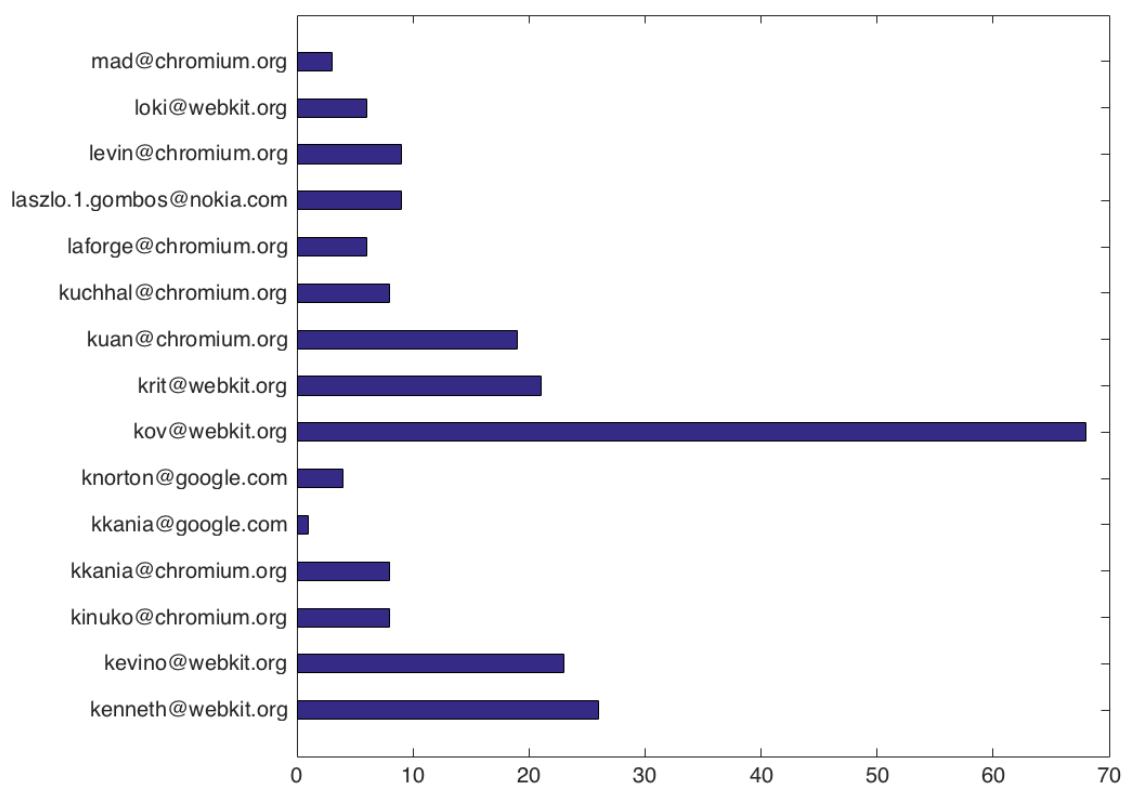


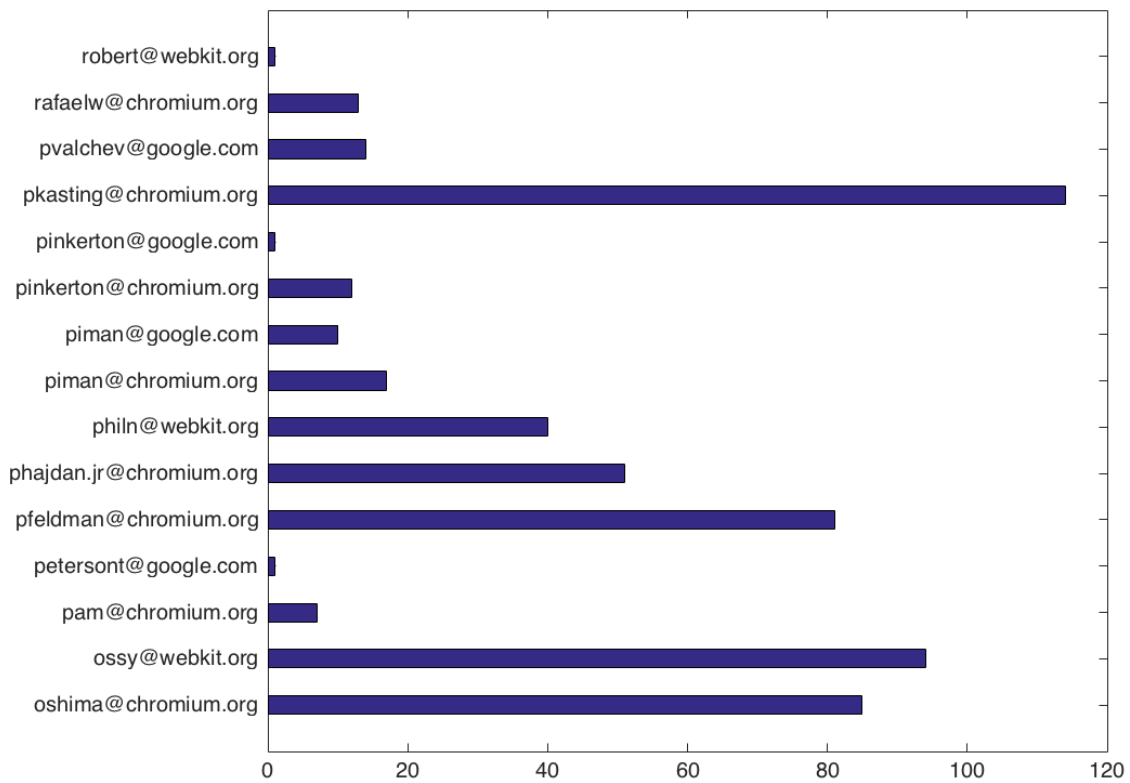
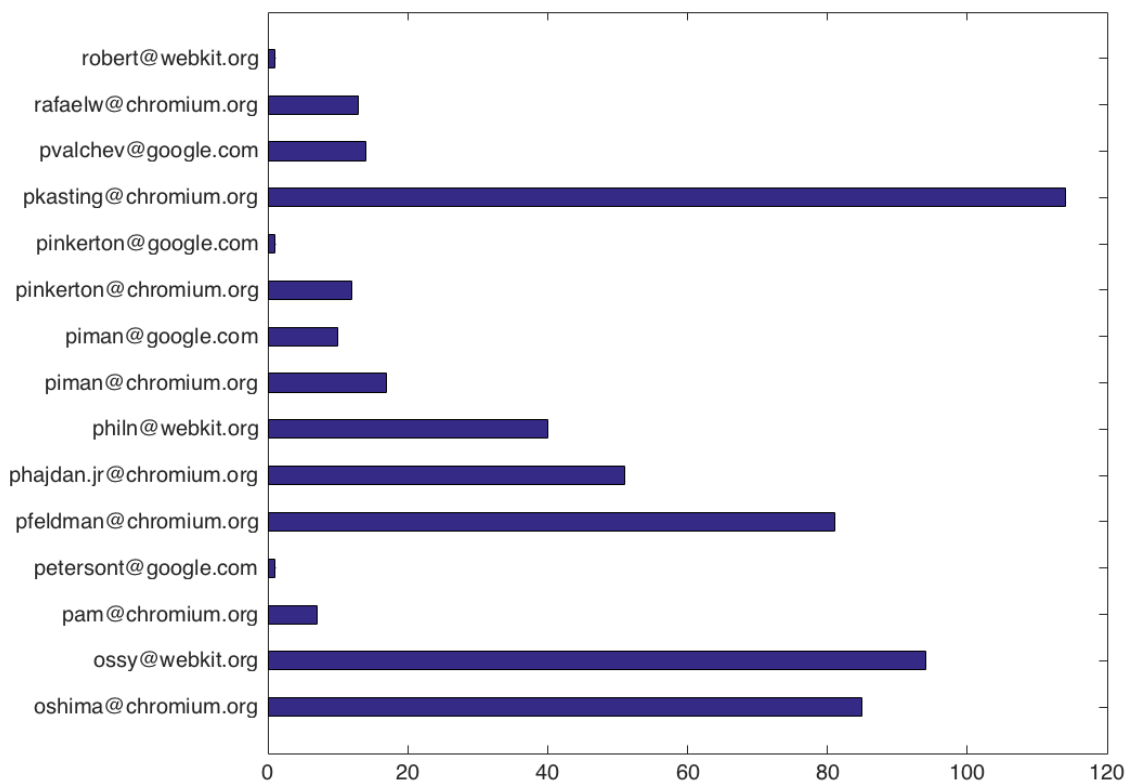


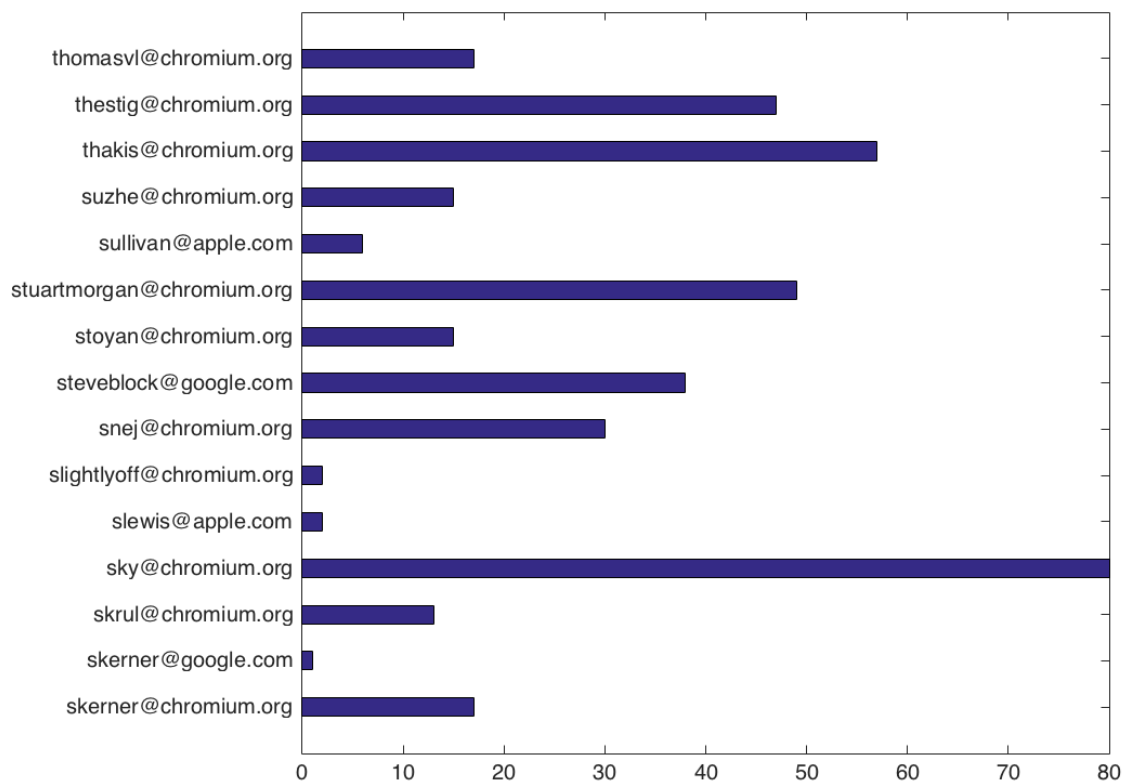
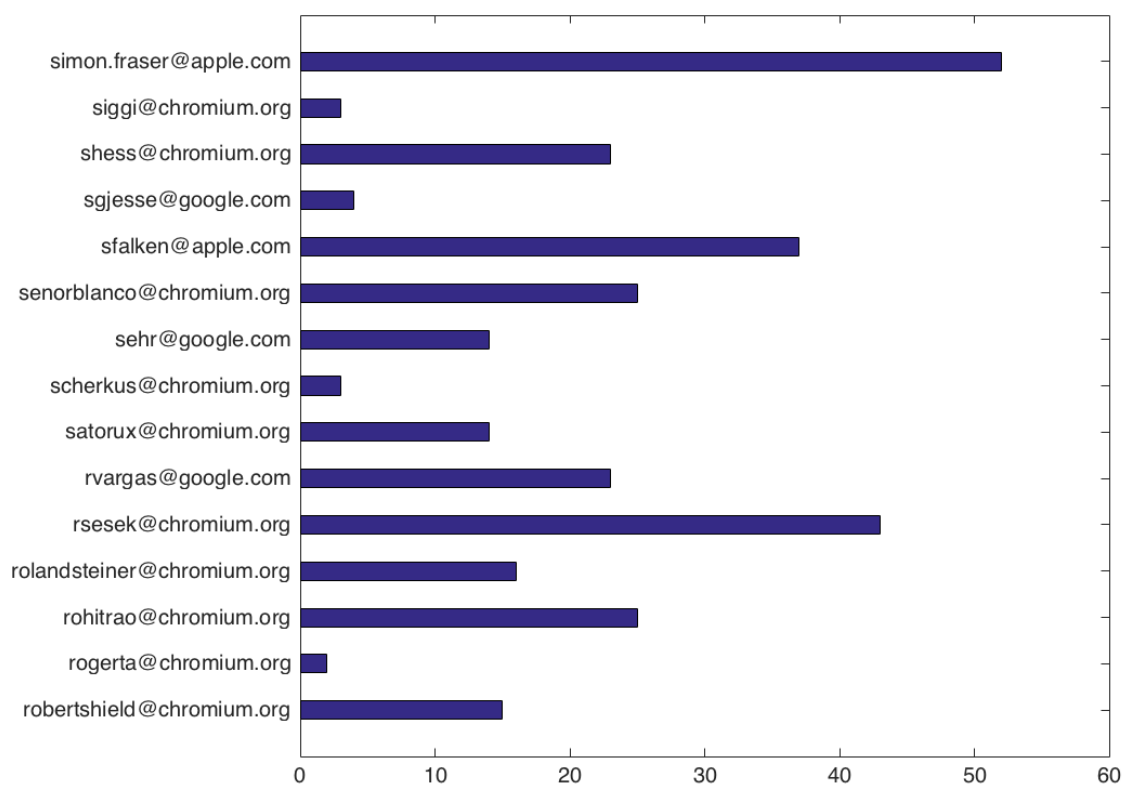


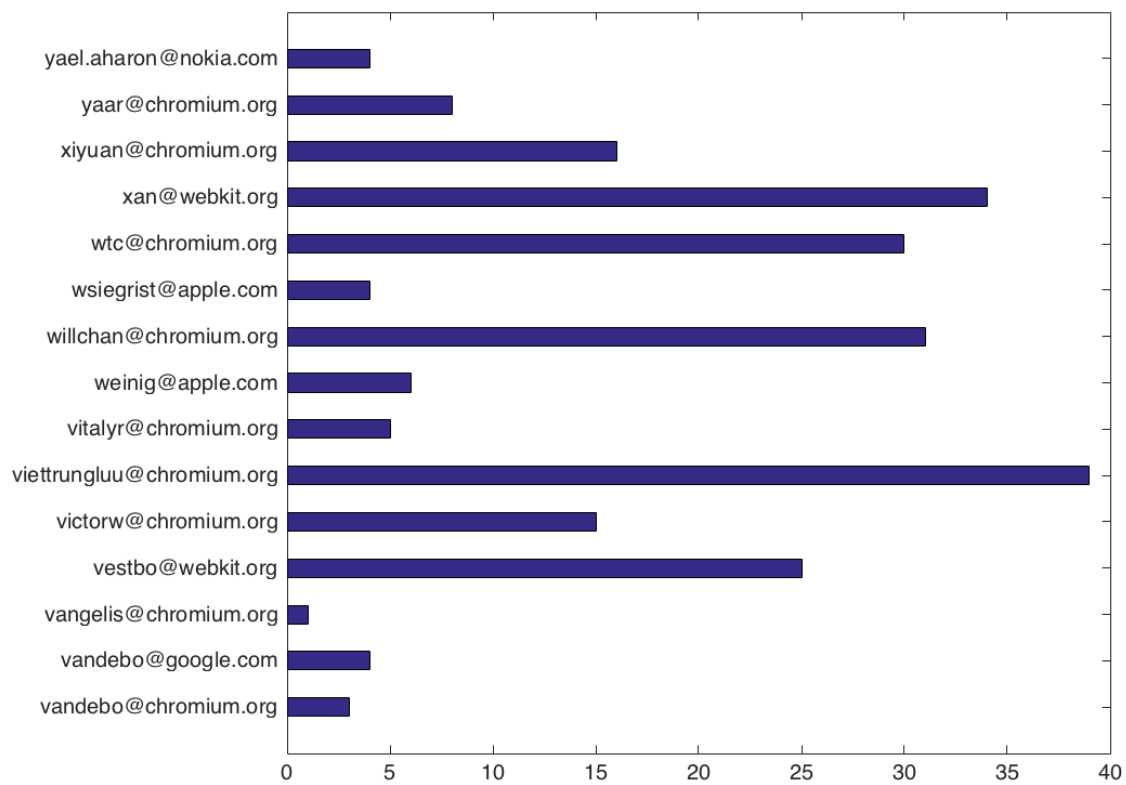
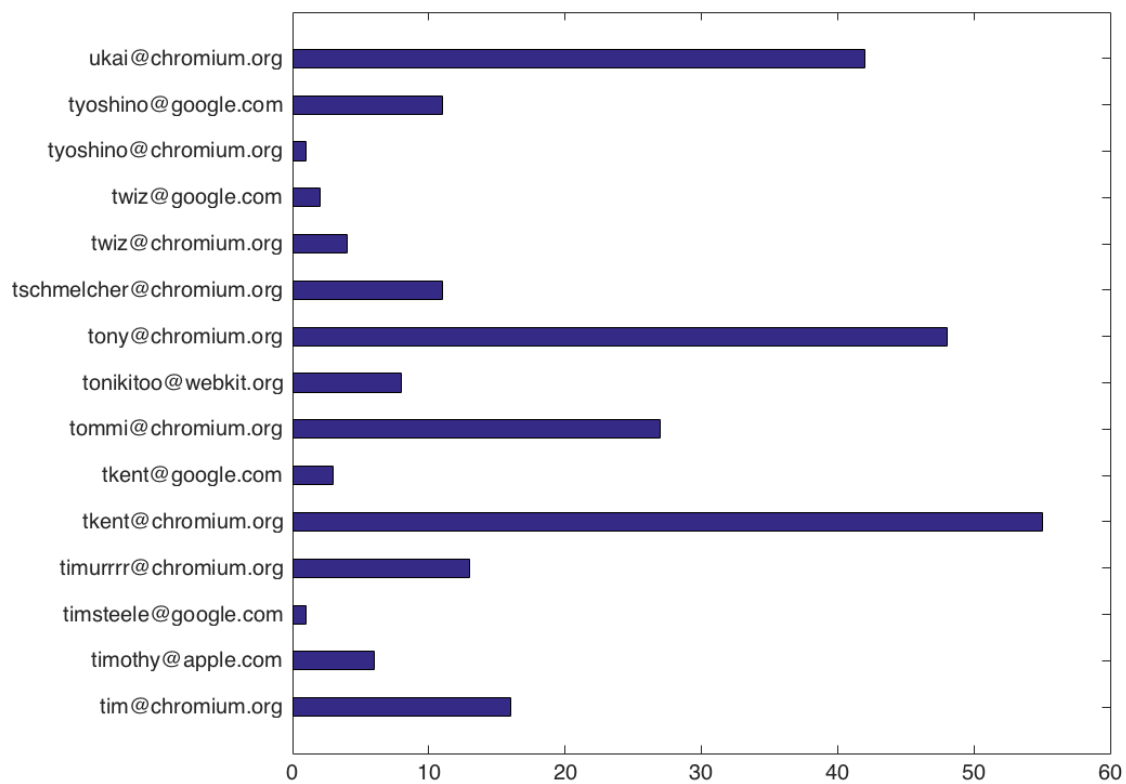


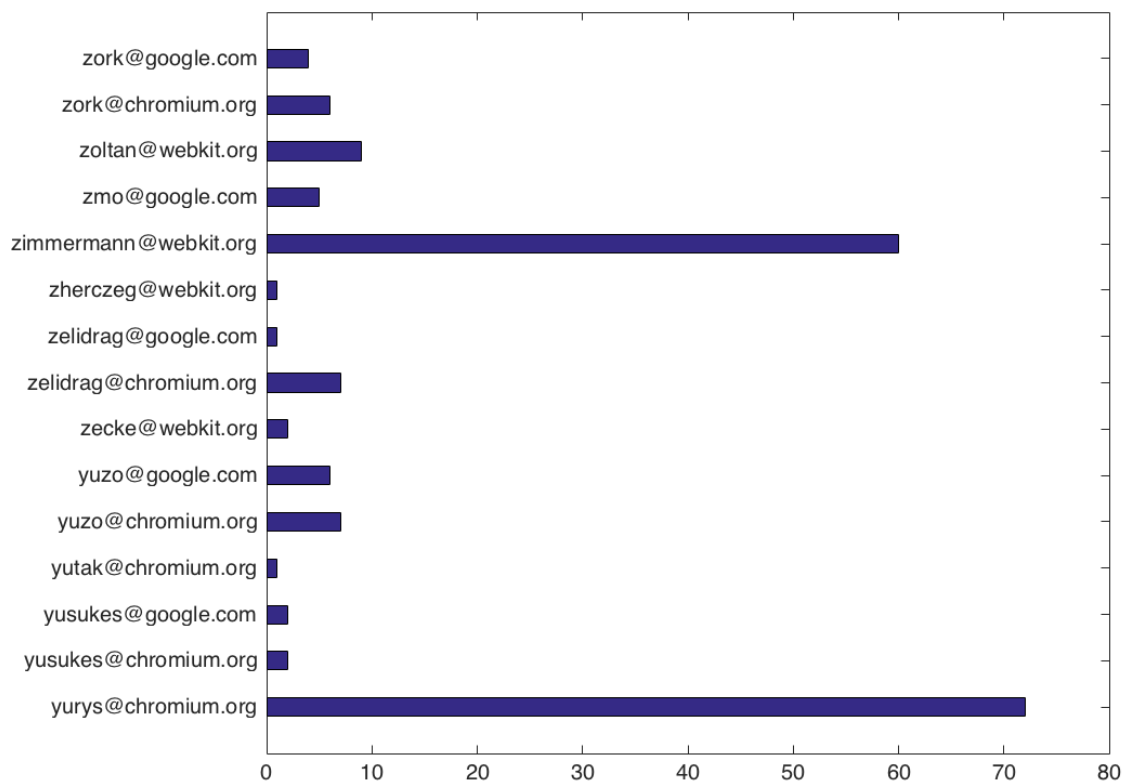












Tables 1-19: The distribution of commit numbers for all developers

3.1.2 Commit frequency of developers

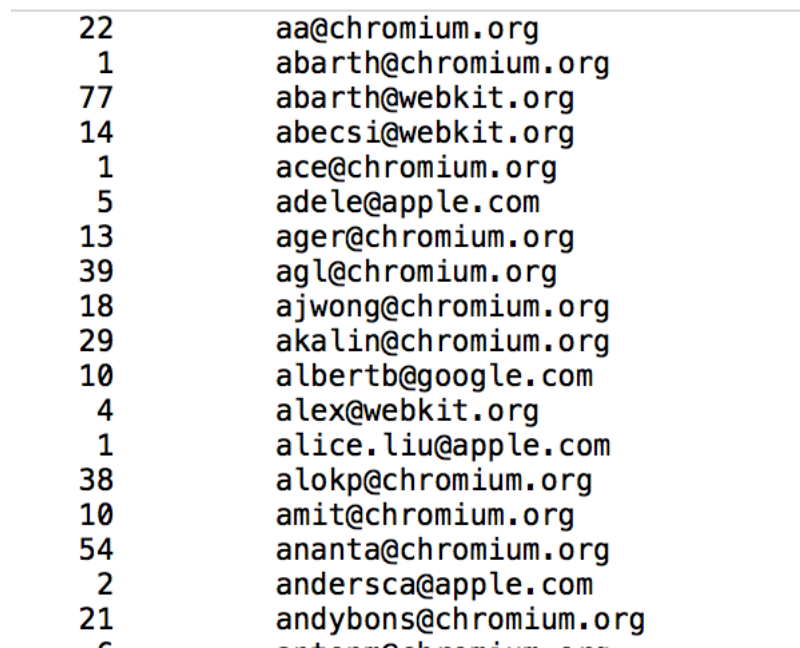
For this aim, we decided to evaluate the frequency of each developer which means the division of number of commits by the active time interval which is difference between first and last commit dates of author. This described process was done with following steps.

Firstly, number of commits for each developer is obtained by using Git command and the result is saved into commitcountandauthors1.log file under the outputs directory.

```
#find commit frequency of committers
echo Frequencies of each developer is calculating..
git shortlog -s --since=2010-01-01 --before=2010-03-01 >outputs/commitcountandauthors1.log
```

Figure 10: Finding number of commits by using Git command

The capture of commitcountandauthors1.log file is given in below screenshot. Note that, the output is alphabetically sorted.



A screenshot of a text file named commitcountandauthors1.log. The file contains a list of email addresses, each preceded by a commit count. The data is sorted alphabetically by email address. The commit counts are: 22, 1, 77, 14, 1, 5, 13, 39, 18, 29, 10, 4, 1, 38, 10, 54, 2, and 21. The email addresses are: aa@chromium.org, abarth@chromium.org, abarth@webkit.org, abecsi@webkit.org, ace@chromium.org, adele@apple.com, ager@chromium.org, agl@chromium.org, ajwong@chromium.org, akalin@chromium.org, albertb@google.com, alex@webkit.org, alice.liu@apple.com, alokp@chromium.org, amit@chromium.org, ananta@chromium.org, andersca@apple.com, and andybons@chromium.org.

22	aa@chromium.org
1	abarth@chromium.org
77	abarth@webkit.org
14	abecsi@webkit.org
1	ace@chromium.org
5	adele@apple.com
13	ager@chromium.org
39	agl@chromium.org
18	ajwong@chromium.org
29	akalin@chromium.org
10	albertb@google.com
4	alex@webkit.org
1	alice.liu@apple.com
38	alokp@chromium.org
10	amit@chromium.org
54	ananta@chromium.org
2	andersca@apple.com
21	andybons@chromium.org

Figure 11:The capture of commitcountandauthors1.log file including commit counts

As a second step, the code snippet named as frequency.c was called to carry out the mentioned division operation as follows.

```
gcc frequency.c -o freq
./freq
```

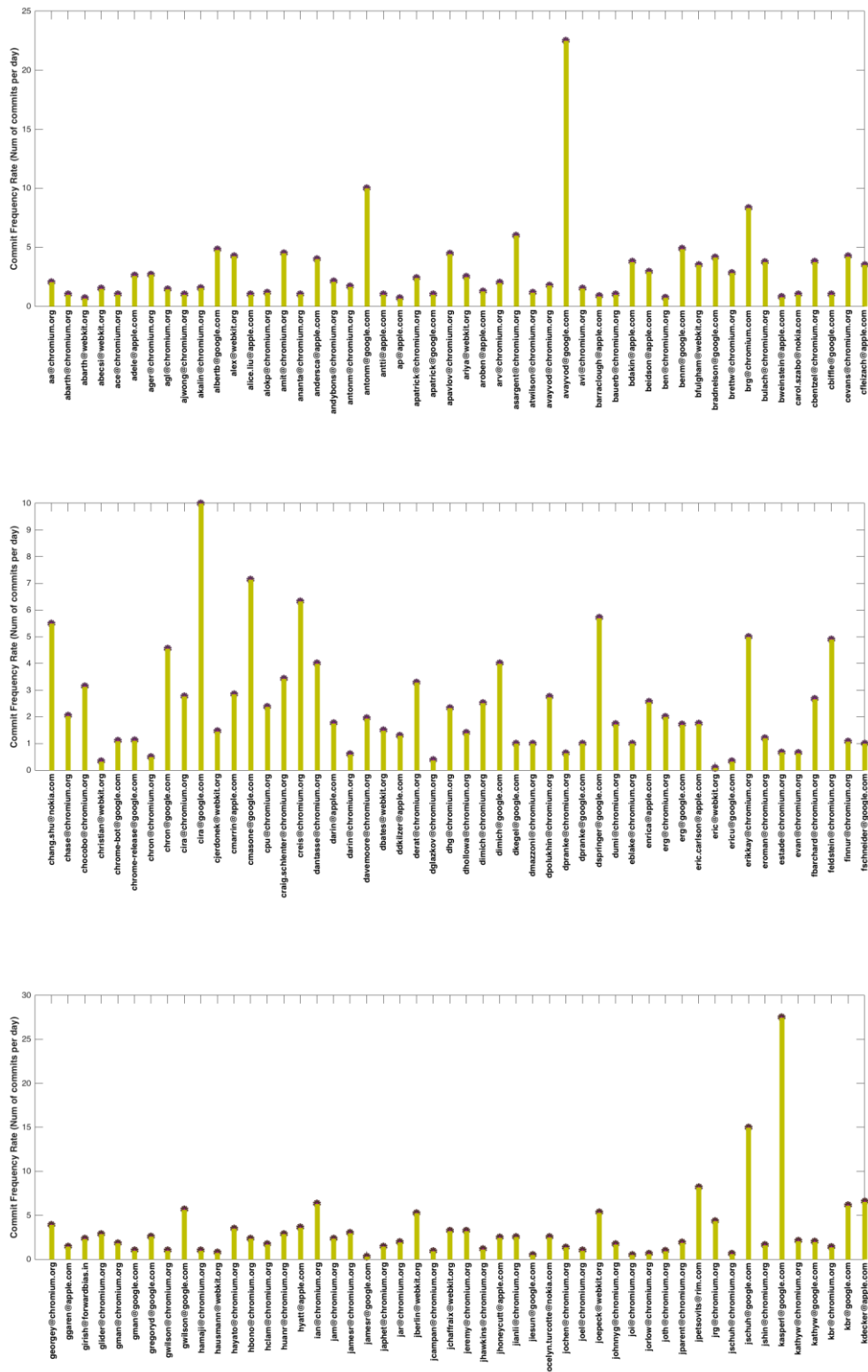
Figure 12: The function call for calculating commit frequency for each developer

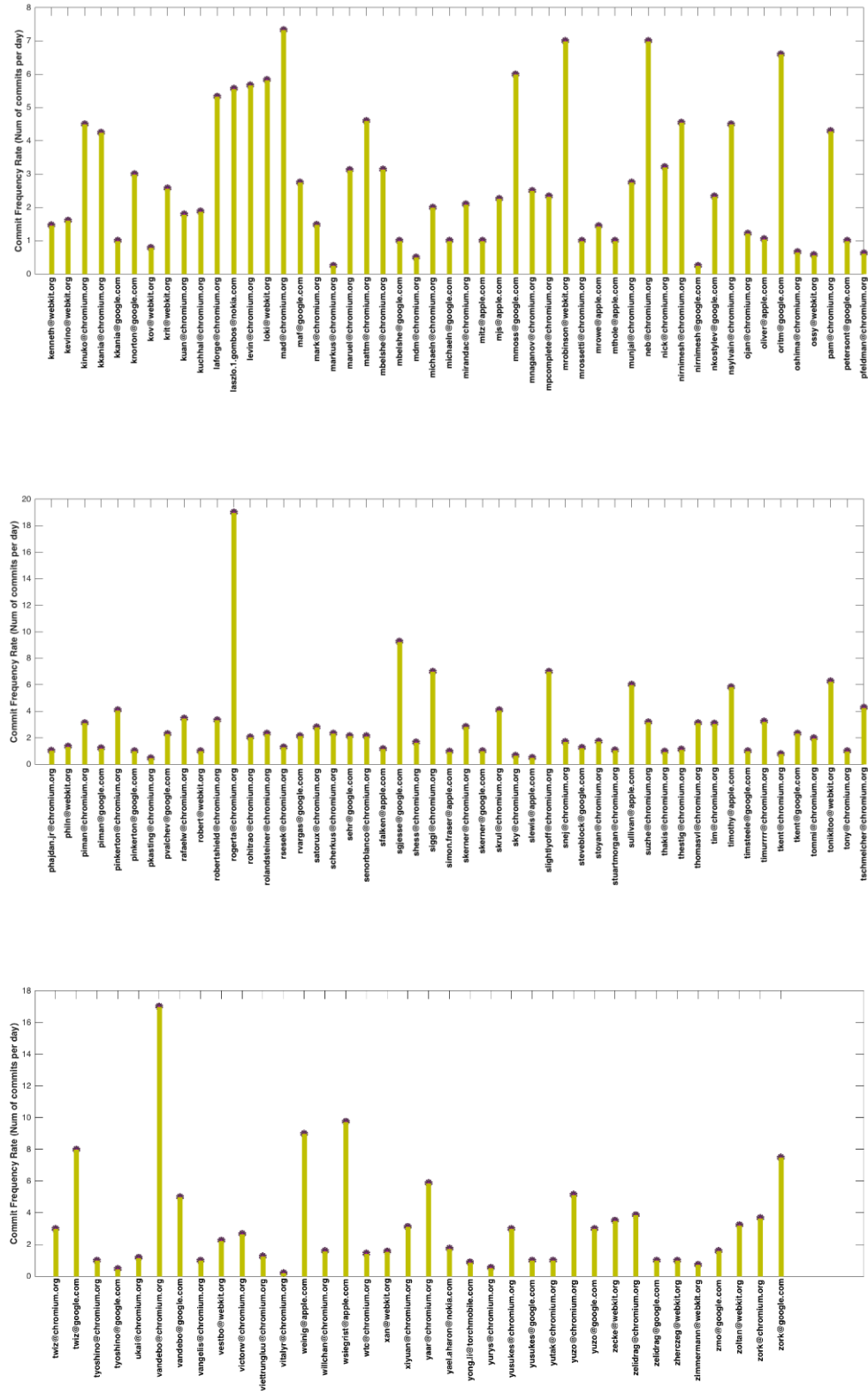
After the function call, commitfrequencyofdevelopers.txt is produced and its content is given in following screenshot.

```
2.05:aa@chromium.org
1.00:abarth@chromium.org
0.69:abarth@webkit.org
1.50:abecsi@webkit.org
1.00:ace@chromium.org
2.60:adele@apple.com
2.69:ager@chromium.org
1.44:agl@chromium.org
1.00:ajwong@chromium.org
1.55:akalin@chromium.org
4.80:albertb@google.com
4.25:alex@webkit.org
1.00:alice.liu@apple.com
1.16:alokp@chromium.org
4.50:amit@chromium.org
1.02:ananta@chromium.org
1.00:andrew@chromium.org
```

Figure 13: The capture of commit frequencies in commitfrequencyofdevelopers.txt

We made visualization of the content of commitfrequencyofdevelopers.txt file and the results are given in below. We used Matlab as a plotting tool.





Tables 20-25: The distribution of commit frequencies for all developers

3.1.3 Identifying top developers who make 80% of the commits

In order to determine top developers of the project we have used following Git command and saved result into commitcountandauthors.log file under the outputs directory. Note that the content is exactly the same with the content of commitcountandauthors1.log file. The only difference comes from the sorting criteria. This time, sorting is done descending number of commits.

```
#find 80% of the commits
echo Top developers are finding..
git shortlog -s -n --since=2010-01-01 --before=2010-03-01 >outputs/commitcountandauthors.log
```

Figure 14: Using Git command for finding top developers

The capture of commitcountandauthors.log file is given in below screenshot. Note that, the output is alphabetically sorted.

623	eric@webkit.org
138	dglazkov@chromium.org
114	pkasting@chromium.org
94	ossy@webkit.org
87	darin@chromium.org
85	oshima@chromium.org
85	evan@chromium.org
85	pfeldman@chromium.org
81	ap@apple.com
80	sky@chromium.org
80	jorlow@chromium.org
79	estade@chromium.org

Figure 15: The capture of commitcountandauthors.log including descending order

Then, we called topdevelopers.c code snippet to carry out mentioned aim. We took into account the committers until the sum of the percentage of the commit of the developers equals at least 80.

```
gcc topdeveloper.c -o topdevelopers
./topdevelopers outputs/commitcountandauthors.log
```

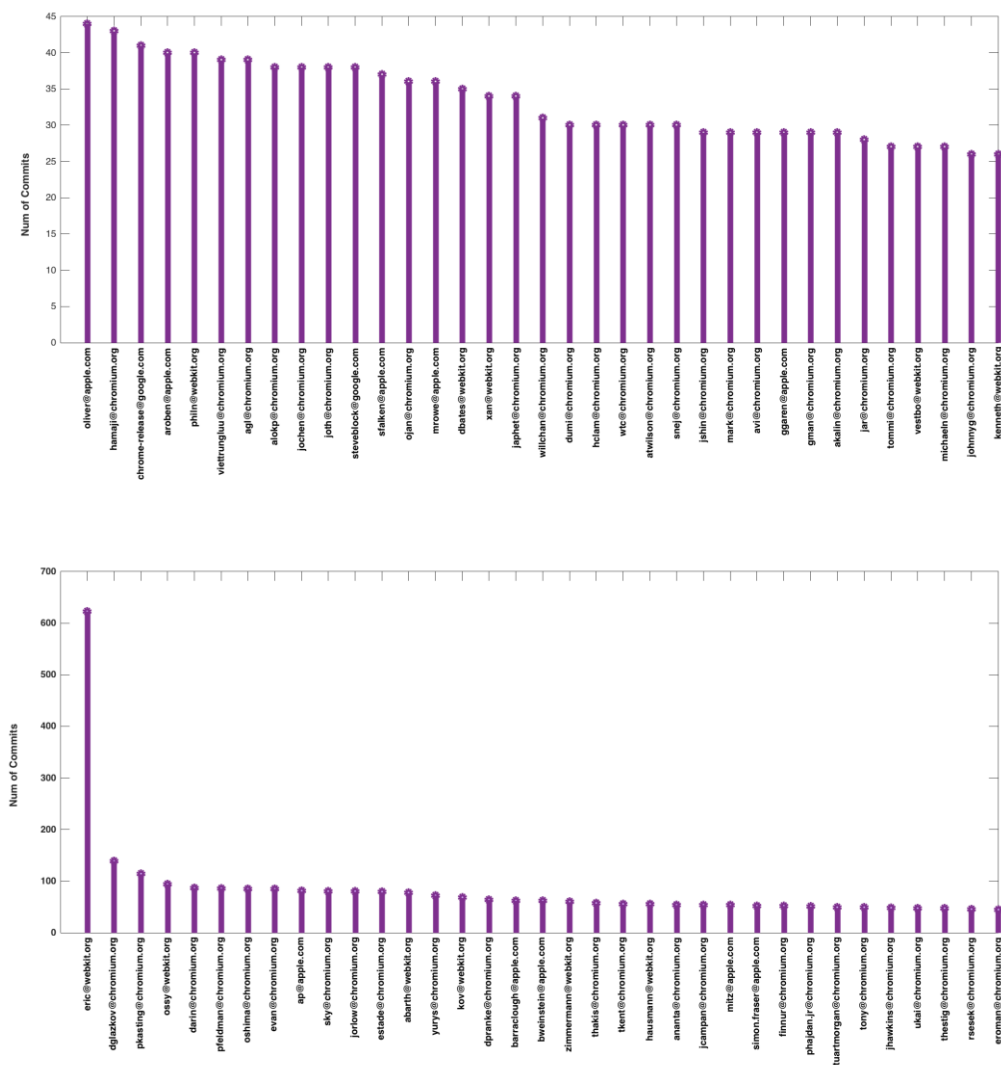
Figure 16: The function call to calculate top developers

The result is saved into topdevelopers.txt with their commit numbers and topdevelopernames.txt with only names. The contents of the files are given below as a screenshot.

623:eric@webkit.org	eric@webkit.org
138:dglazkov@chromium.org	dglazkov@chromium.org
114:pkasting@chromium.org	pkasting@chromium.org
94:ossy@webkit.org	ossy@webkit.org
87:darin@chromium.org	darin@chromium.org
85:oshima@chromium.org	oshima@chromium.org
81:ap@apple.com	ap@apple.com
81:pfeldman@chromium.org	pfeldman@chromium.org
81:evan@chromium.org	evan@chromium.org
80:sky@chromium.org	sky@chromium.org
80:jorlow@chromium.org	jorlow@chromium.org
79:estade@chromium.org	estade@chromium.org
77:abarth@webkit.org	abarth@webkit.org
72:yurys@chromium.org	yurys@chromium.org
69:koy@webkit.org	koy@webkit.org

Figure 17: The capture of topdevelopers.txt including developers with commit nums

The results are visualized as following screenshots:



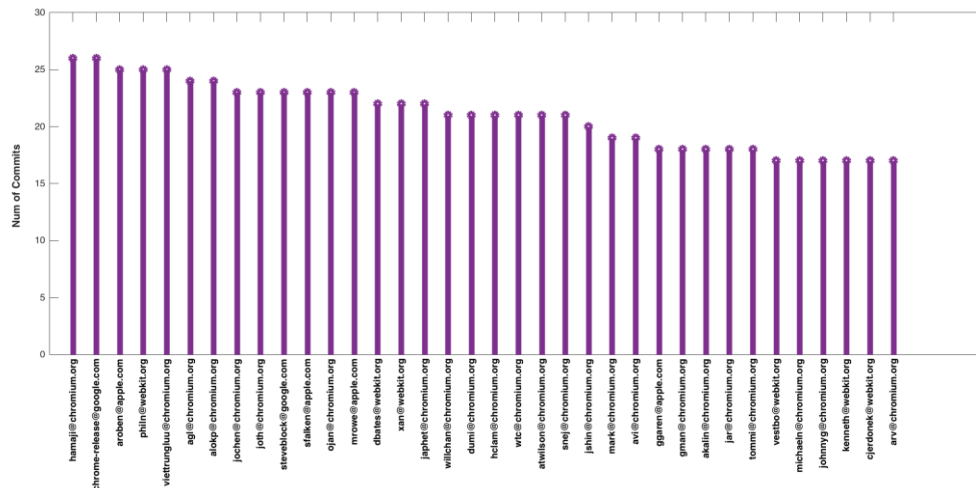


Table 26-28: The plot of top developers with commit numbers

3.2 Identifying Edited File Sets in Commits

3.2.1 Creating matrix

In this section, we created a matrix in each rows represent files and columns represent developers. If the index at (i,j) in matrix is 1, it should indicate 1 if file i is committed by developer j, or 0 otherwise.

For this aim, the following code is written.

```
#Part3: Generation of matrix
for e in "${authors[@]}"
do
echo "*" >>outputs/allchangedfilesandauthors.log
echo "$e" >>outputs/allchangedfilesandauthors.log
git log --since=2010-01-01 --before=2010-03-01 --pretty="%H" --author="$e" | while read commit_hash; do git
show --oneline --name-only $commit_hash | tail -n+2; done | sort | uniq >>outputs/
allchangedfilesandauthors.log
echo "*" >>outputs/allchangedfilesandauthors.log
done
```

Figure 18: The code to generate the files/developers matrix

In the above Bash script, we specify a format as following:

```
*  
  
author name  
edited files of the author  
  
**
```

The results are saved into allchangedfilesandauthors.log file and its content is given below:

```
*  
zmo@google.com  
o3d/core/win/d3d9/renderer_d3d9.cc  
o3d/core/win/d3d9/renderer_d3d9.h  
o3d/plugin/win/main_win.cc  
**  
*  
zoltan@webkit.org  
third_party/WebKit/JavaScriptCore/ChangeLog  
third_party/WebKit/JavaScriptCore/wtf/FastMalloc.cpp  
third_party/WebKit/WebCore/ChangeLog  
third_party/WebKit/WebCore/dom/XMLTokenizerQt.cpp  
third_party/WebKit/WebCore/loader/icon/IconDatabaseClient.h  
third_party/WebKit/WebCore/platform/graphics/qt/FontPlatformData.h  
third_party/WebKit/WebCore/platform/graphics/qt/GraphicsContextQt.cpp  
third_party/WebKit/WebCore/websockets/WorkerThreadableWebSocketChannel.h  
**  
*  
zork@chromium.org  
build/install-build-deps.sh  
chrome/browser/sync/engine/idle_query_linux.cc
```

Figure 19: The capture of allchangedfilesandauthors.log file

As a second step, we listed all the files in the repo and saved results into allfiles.log file.

```
git ls-files >outputs/allfiles.log
```

Figure 20: The Git command to list all the files in the repo and save into a file

The content of the allfiles.log is given below

```

ui/views/masked_targeter_delegate.h
ui/views/metrics.cc
ui/views/metrics.h
ui/views/metrics_aura.cc
ui/views/metrics_mac.cc
ui/views/mouse_constants.h
ui/views/mouse_watcher.cc
ui/views/mouse_watcher.h
ui/views/mouse_watcher_view_host.cc
ui/views/mouse_watcher_view_host.h
ui/views/mus/BUILD.gn
ui/views/mus/DEPS
ui/views/mus/aura_init.cc
ui/views/mus/aura_init.h
ui/views/mus/display_converter.cc
ui/views/mus/display_converter.h

```

Figure 21: The capture of allfiles.log including all files in the repo

Then, the result is used into preprocess_matrix.cpp code. In the code, we made little modification by taking only the files that changed by top developers. Then, the result is saved into files.txt file into outputs directory.

```

echo Matrix is generating...
g++ preprocess_matrix.cpp -o pre
./pre outputs/allchangedfilesandauthors.log

```

Figure 22: The C++ function call to preprocess the matrix

Note that, files.txt can contain the same file more than one. Assume that developer A and developer B edited the x.txt file. In files.txt file x.txt occurs twice so that we used

```
sort -u outputs/files.txt >outputs/uniqfiles.txt
```

command to avoid duplicate lines and as can be seen above the result is saved into uniqfiles.txt.

As a last step, we called creatematrix.cpp file with following parameter. The result is saved into matrix.txt file under the outputs directory.

```

g++ creatematrix.cpp -o matrix
./matrix outputs/allchangedfilesandauthors.log

```

Figure 23: The C++ function call to create the matrix

The content of the matrix.txt file is given below.

Figure 24: The capture of the matrix including files/developers info

3.3 Building and Visualization of Socio-technical Network of Chrome Project

Firstly, we saved the matrix.txt in excel file format and used the “one mode network analysis” feature. In our matrix, the rows represent the files and the columns represent the files. Here, the meaning of the matrix is that if or not the developers work on the files. If a developer works on a file, then in that cell of the matrix “1” is written, and vice versa “0” is written. With this file format, we can analyze “two mode network analysis”. In this way, we can see the files as one type of nodes and the developers as another type of nodes. Furthermore, the edges are out of the developers to files. However, since we want to see the relations between the developers not the files and the developers, we used “Convert two mode to one mode data” feature in UCINET 6. Because the developers are the columns in the matrix (i.e. in the two mode data), we chose the “columns” for the mode in the tool. Thus, we can analyze the relations between the developers by means of collaboration on the files.

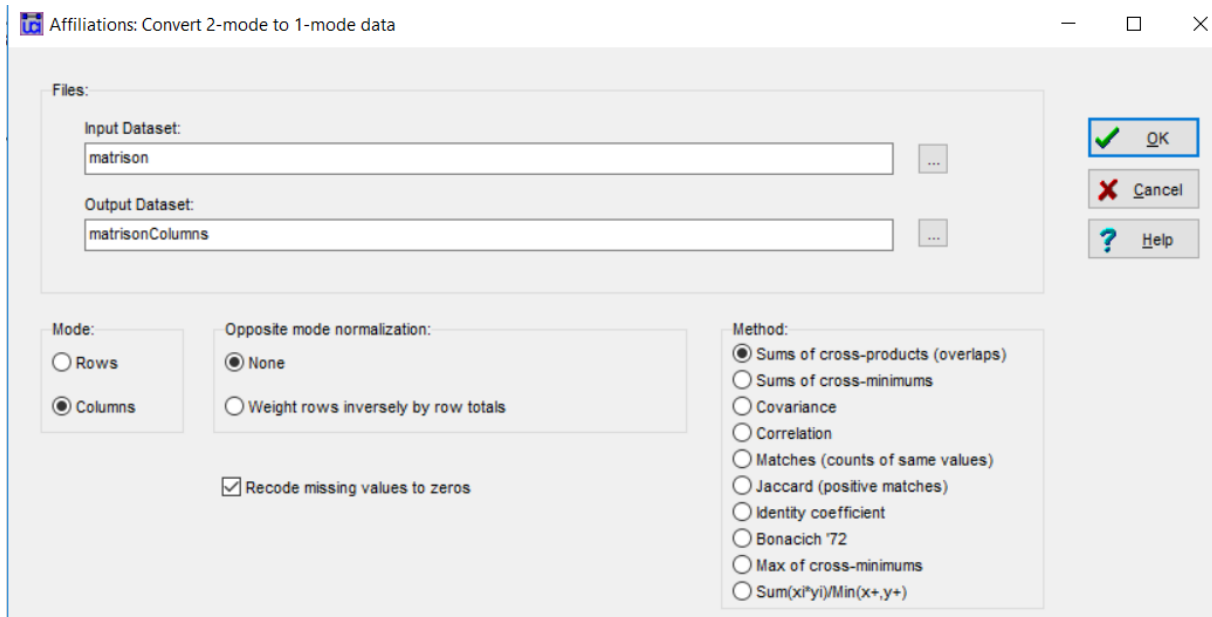


Figure 27: The window of “Convert two mode to one mode data” in UCINET 6

After obtaining the one mode data, we used NetDraw tool of UCINET 6 so as to visualize the collaboration of the developers:

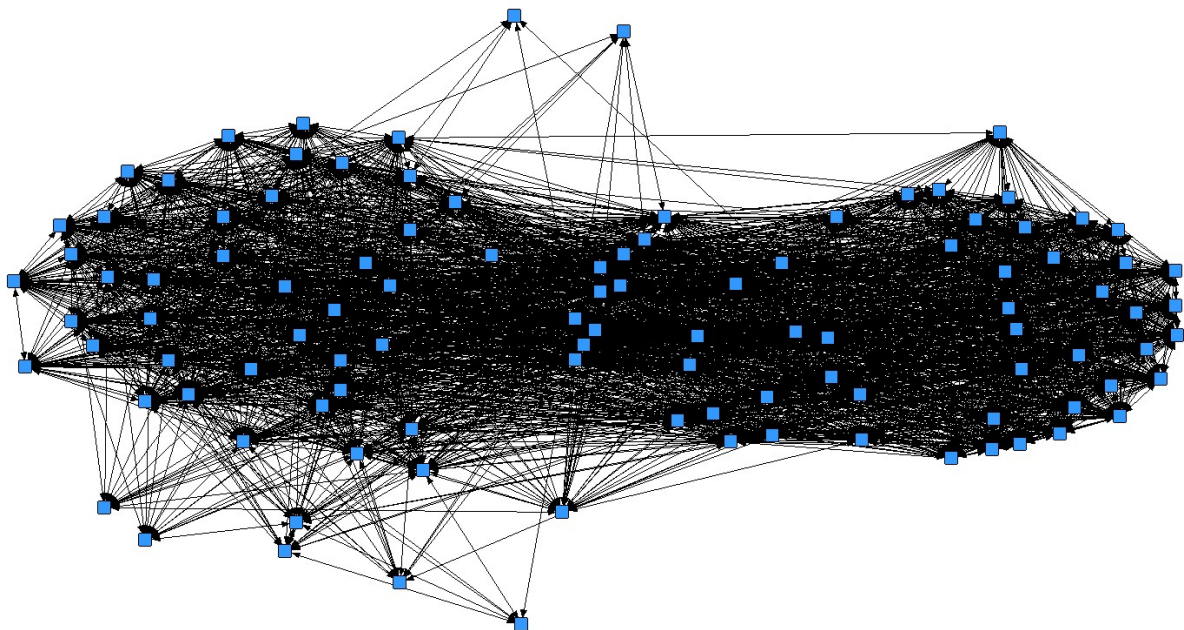


Figure 25: The social network analysis graph for developers done by UCINET tool

4. Sample Execution Result

Note that all the written code for the project is independent from the project which means it can be executed for any project. For instance, the code is executed for a random project named as Gitmagic. It can be found with a repo <https://github.com/blynn/gitmagic.git>. The result of the execution is given below.

```
Nurefsan-MacBook-Pro:gitmagic nurefsansertbas$ date
Fri May 20 02:53:36 EEST 2016
Nurefsan-MacBook-Pro:gitmagic nurefsansertbas$ ./son.sh
Authors are extracting..
Total commits with commit dates are extracting..
Extracting data of given time interval..
Result is saved into outputs/commitof2months_c1.txt and outputs/commitof2months_c2.txt

Frequencies of each developer is calculating..
Result is saved into commitfrequencyofdevelopers.txt under outputs directory

Top developers are finding..
Note: Top developer num of commits should be more than 28 out of 35 commits
There are 4 top developer
Top developers are found and written into topdevelopernames.txt under outputs directory

Matrix is generating...
Result is saved into matrix.txt under outputs directory

Cleaning operations id doing
Nurefsan-MacBook-Pro:gitmagic nurefsansertbas$ date
Fri May 20 02:53:43 EEST 2016
Nurefsan-MacBook-Pro:gitmagic nurefsansertbas$ █
```

Figure 26: The sample execution result for the project

It may take less than 1 minute. However, due to the size of the Chromium project the execution time of the project is approximately 2 hours depending on the machine.

5. References

- [1]** Blynn/gitmagic. (n.d.). Retrieved May 21, 2016, from <https://github.com/blynn/gitmagic.git>
- [2]** What is version control? (n.d.). Retrieved May 21, 2016, from <https://www.atlassian.com/git/tutorials/what-is-version-control/>
Git. (n.d.). Retrieved May 21, 2016, from <https://git-scm.com/>
- [3]** Git. (n.d.). Retrieved May 21, 2016, from <https://git-scm.com/>
- [4]** Data Mining: What is Data Mining? (n.d.). Retrieved May 21, 2016, from <http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm>
- [5]** The Chromium Projects. (n.d.). Retrieved May 21, 2016, from <https://www.chromium.org/>