MINING CHROME REPOSITORY

BLG 440E Computer Project 2

040110078
Nurefşan Sertbaş

040090508
Betül Kantepe

Project #3 – Group 16

Purpose of the Project

Goal is to identify social and technical dependencies in Chrome project

PROJECT DESCRIPTION

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.

WHAT IS DATA MINING?

THE PROCESS

- PROVIDES RETRIEVING OF HIDDEN USEFUL INFORMATION FROM LARGE DATABASES.

Data mining approach used in the project

We used data mining approach so as to mine Chromium's version control system by writing scripts.

We followed below steps:

- ❑ IDENTIFY TOP DEVELOPERS
- ❑ IDENTIFY EDITED FILE SETS
- ❑ BUILD SOCIAL NETWORK

About the chromium project

- Analyze by specifying time interval
- ➢ instead of choosing specific directory
- more logical to take into account whole development of the project.
- ➢ The outputs provide us a general perspective about project development history.

WHAT IS VERSION CONTROL SYSTEM?

- 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.

WHAT IS GIT?

- One of the commonly used VCS tools today
- Similar to other popular VCS systems
- Git is free and open source.

Writing scripts by using git command

Several ways to use Git commands

- ❖ Graphical user interface
- ❖ Command line

✓ We preferred to use Git from command line

✓ We wrote our scripts by using Bash scripting language with using directly outputs of the Git commands.

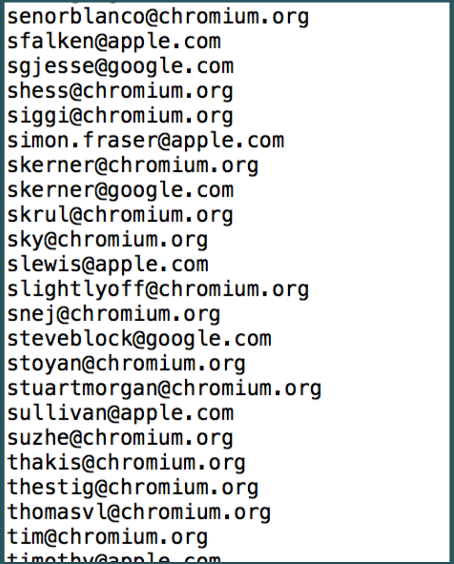
✓ We benefit from C and C++ function calls in the script

MINING INFORMATION

- We mined 5984 commits between dates 01- 01-2010 and 01-03-2010
- ➢ two months.

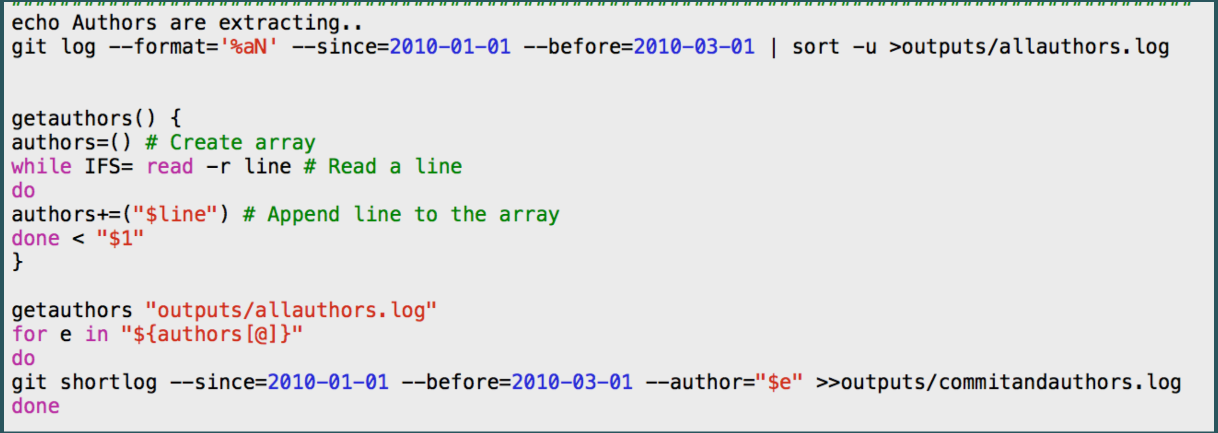
Data mining process

- ✓ We extract all the authors which are actively committed in chosen time range.



```
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
```

Git command used



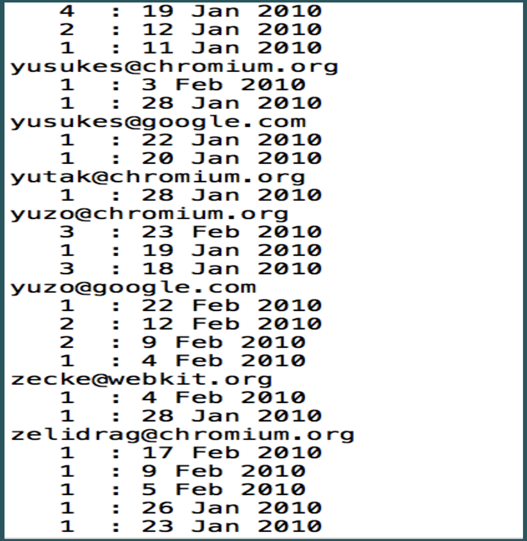
```
echo Authors are extracting..
git log --format='%aN' --since=2010-01-01 --before=2010-03-01 | sort -u >outputs/allauthors.log


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
```

- Extracting and saving all authors of the project

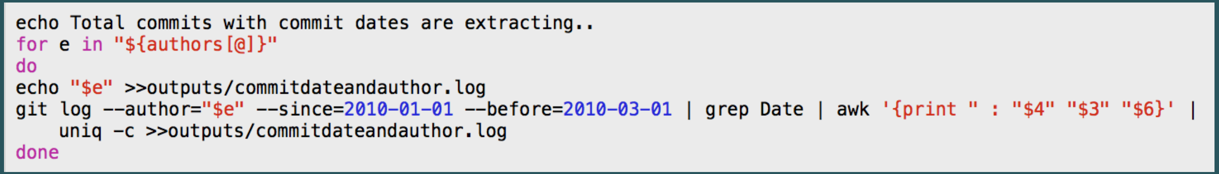
Data mining process



```
     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
```

- ✓ Secondly, we extracted all the commits and commit dates for each committers.

Git command used

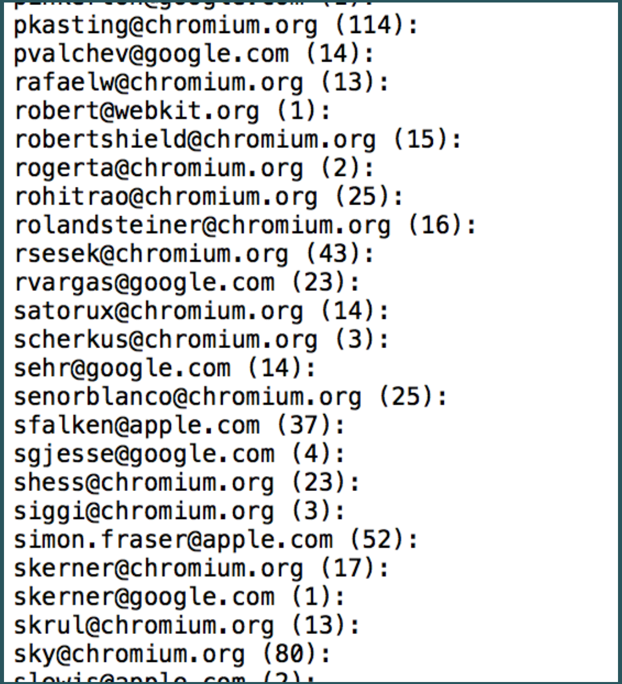


```
echo Total commits with commit dates are extracting..
for e in "${authors[@]}"
do
echo "$e" >>outputs/commitdateandauthor.log
git log --author="$e" --since=2010-01-01 --before=2010-03-01 | grep Date | awk '{print " : "$4" "$3" "$6}' |
    uniq -c >>outputs/commitdateandauthor.log
done
```

- Extracting all commits and commit dates for each developer

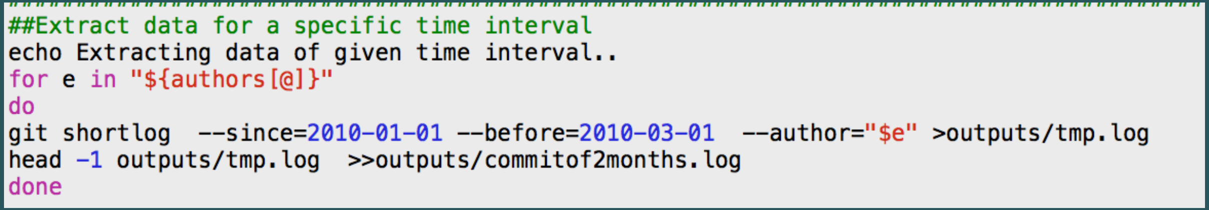
Data mining process

- ✓ As a last step of the preprocess, we extract how many commits are done in selected time interval by each author



```
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):
```

Git command used

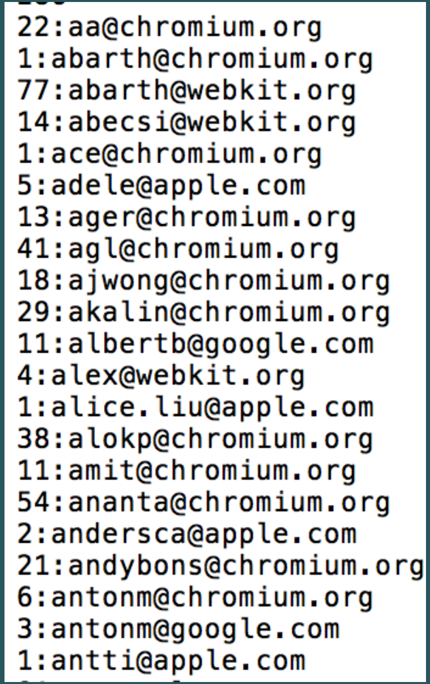


```
##Extract data for a specific time interval
echo Extracting data of given time interval..
for e in "${authors[@]}"
do
git shortlog  --since=2010-01-01 --before=2010-03-01  --author="$e" >outputs/tmp.log
head -1 outputs/tmp.log  >>outputs/commitof2months.log
done
```

- Extracting the number of commits in a specific time interval

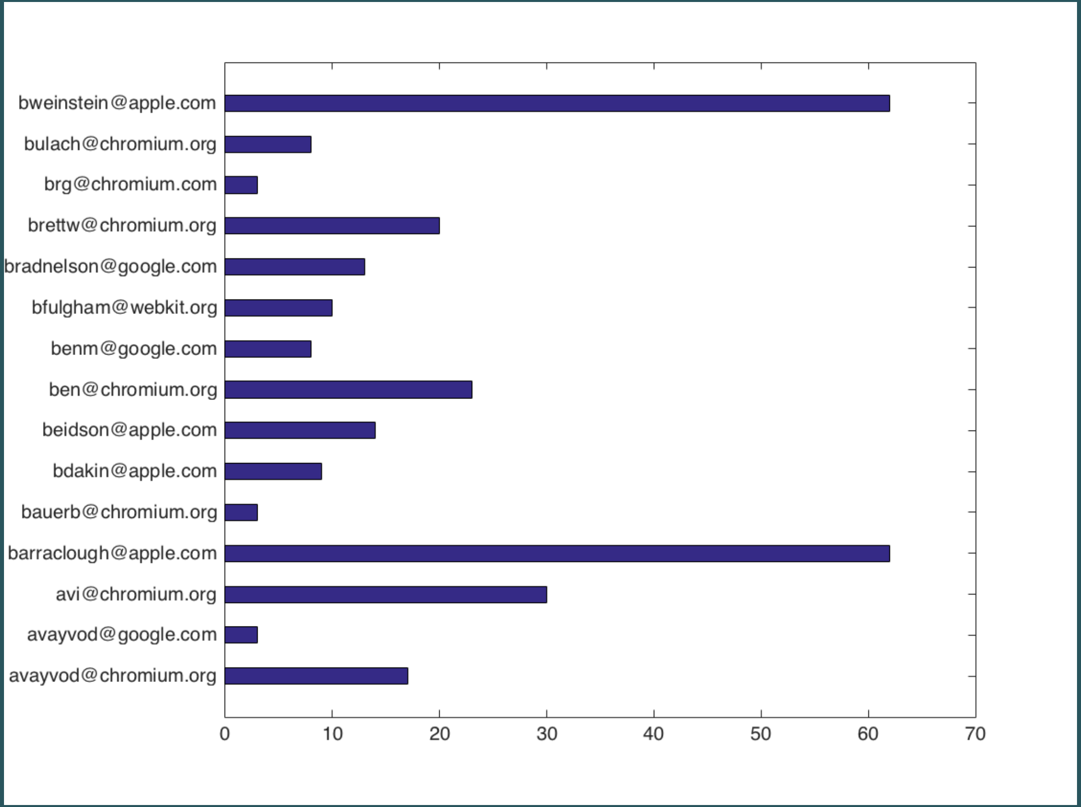
The distribution of commits of developers

- We benefit from code snippet written in C
  - Calculates number of commits of each author in selected time interval.



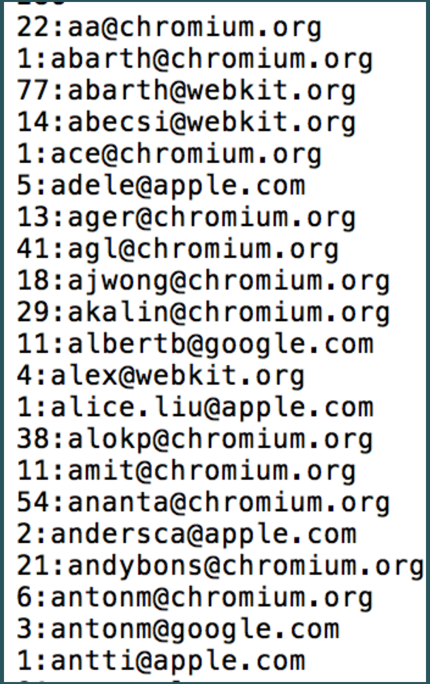
```
22:aa@chromium.org
1:abarth@chromium.org
77:abarth@webkit.org
14:abecsi@webkit.org
1:ace@chromium.org
5:adele@apple.com
13:ager@chromium.org
41:agl@chromium.org
18:ajwong@chromium.org
29:akalin@chromium.org
11:albertb@google.com
4:alex@webkit.org
1:alice.liu@apple.com
38:alokp@chromium.org
11:amit@chromium.org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

Plots of all developers In terms of commit numbers



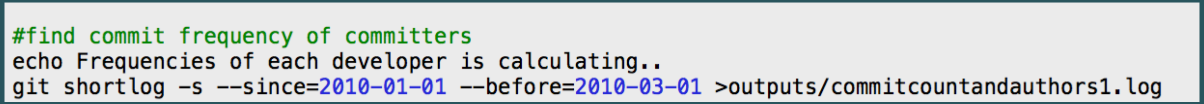
Calculating commit frequency of developers

- ✓ The frequency of each developer means the division of number of commits by the active time interval which is difference between first and last commit dates of author.
- ✓ Firstly, number of commits for each developer is obtained by using Git command.



```
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
```

Git command used



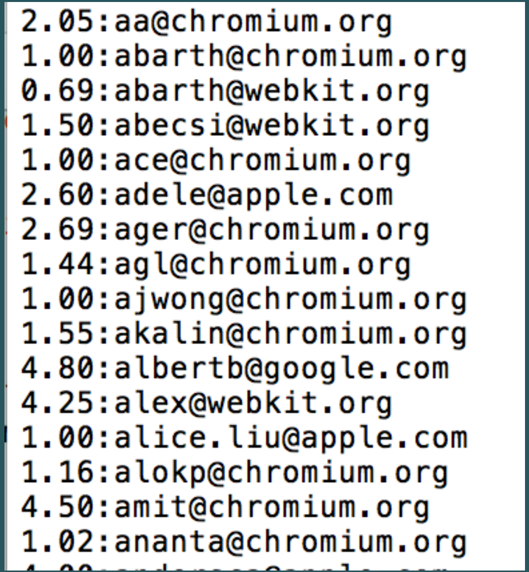
```
#find commit frequency of committers
echo Frequencies of each developer is calculating..
git shortlog -s --since=2010-01-01 --before=2010-03-01 >outputs/commitcountandauthors1.log
```

❖ Finding number of commits by using Git command

Calculating commit frequency of developers

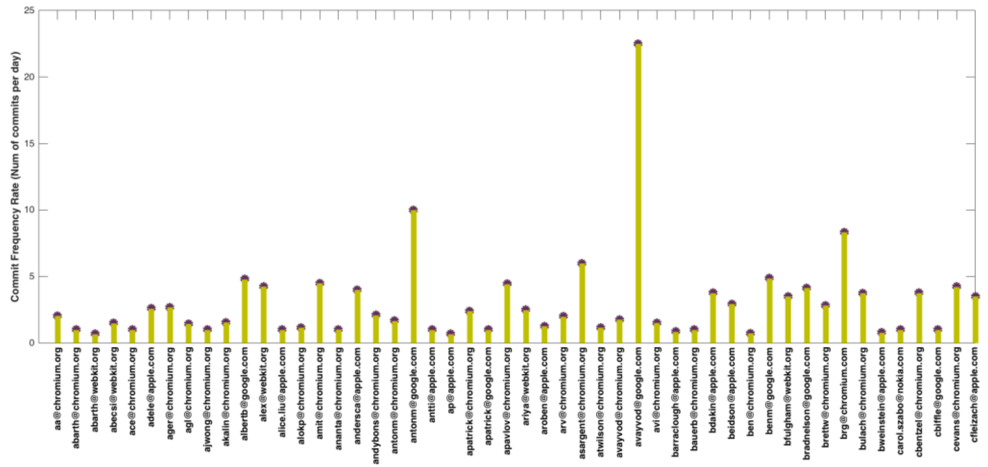
- ✓ As a second step, the code snippet named as frequency.c was called to carry out the mentioned division operation

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

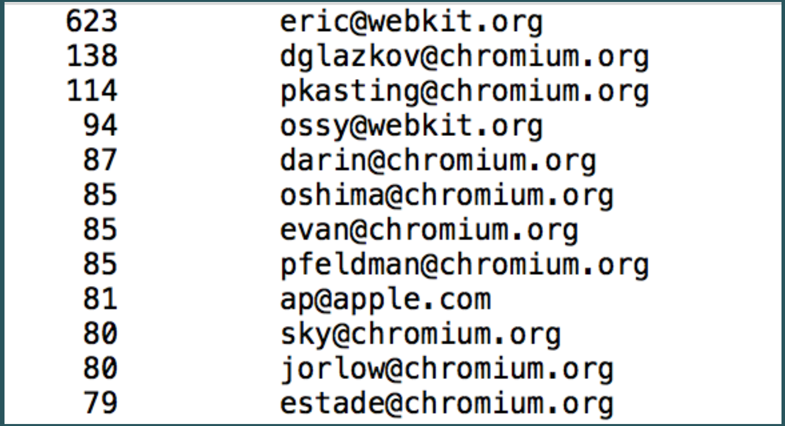


```
2.05:aa@chromium.org
1.00:abarth@chromium.org
0.69:abarth@webkit.org
1.50:abecsi@webkit.org
1.00:ace@chromium.org
2.60:adele@apple.com
2.69:ager@chromium.org
1.44:agl@chromium.org
1.00:ajwong@chromium.org
1.55:akalin@chromium.org
4.80:albertb@google.com
4.25:alex@webkit.org
1.00:alice.liu@apple.com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

Plots of developers In terms of commit frequency

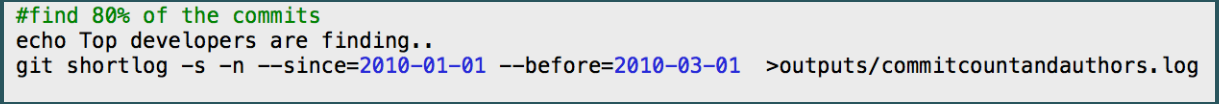


FINDING TOP DEVELOPERS MAKING %80 OF COMMITS



```
   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
```

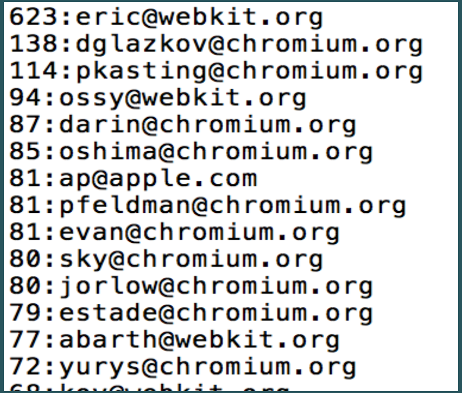
- ✓ WE HAVE USED FOLLOWING GIT COMMAND AND SAVED RESULT INTO COMMITCOUNTANDAUTHORS.LOG FILE



```
#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
```

FINDING TOP DEVELOPERS MAKING %80 OF COMMITS

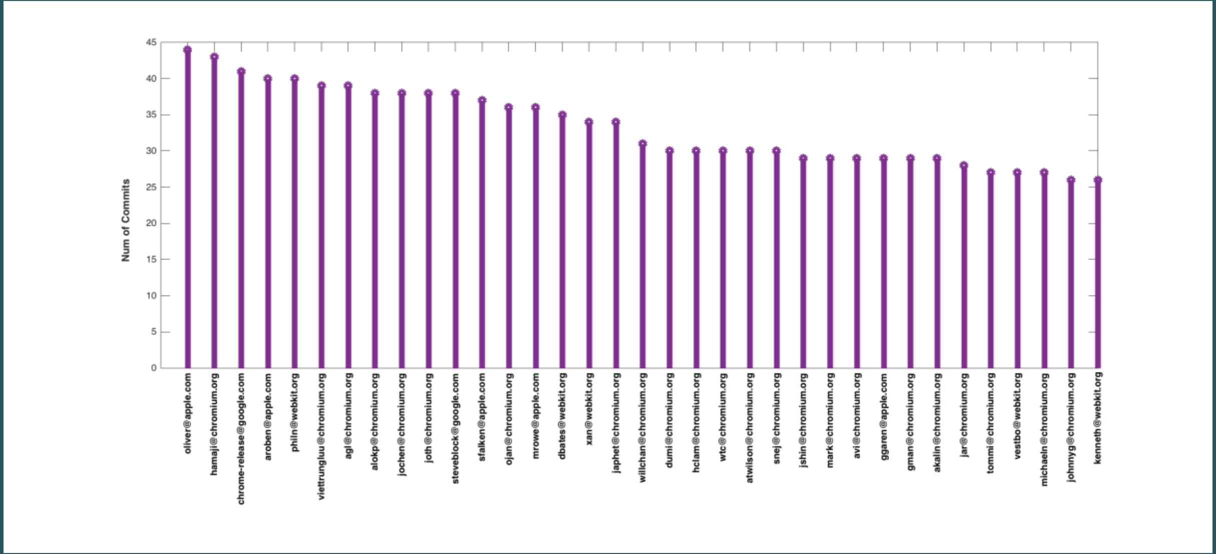
- ✓ THEN, WE CALLED TOPDEVELOPERS.C CODE SNIPPET.
- ➢ THE COMMITTERS UNTIL THE SUM OF THE PERCENTAGE OF THE COMMIT OF THE DEVELOPERS EQUALS AT LEAST 80.



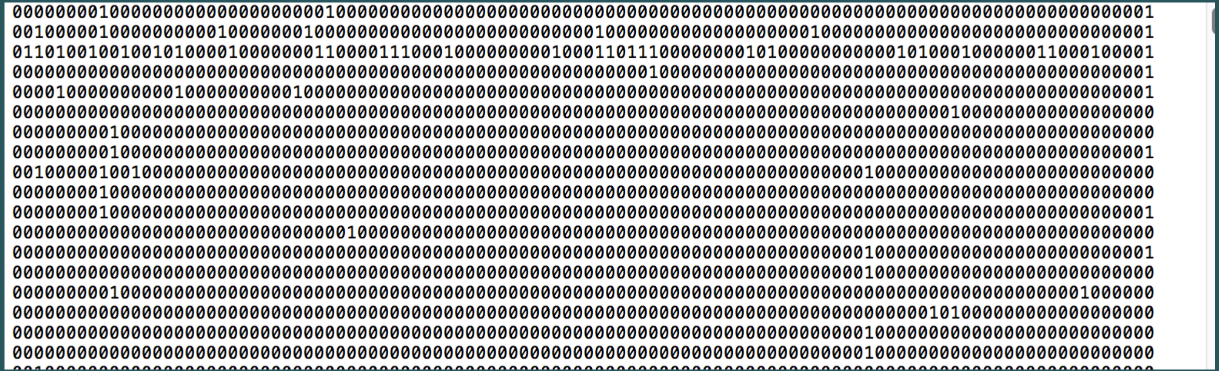
```
623:eric@webkit.org
138:dglazkov@chromium.org
114:pkasting@chromium.org
94:ossy@webkit.org
87:darin@chromium.org
85:oshima@chromium.org
81:ap@apple.com
81:pfeldman@chromium.org
81:evan@chromium.org
80:sky@chromium.org
80:jorlow@chromium.org
79:estade@chromium.org
77:abarth@webkit.org
72:yurys@chromium.org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

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

Plots of top developers Make %80 of commits



Creating Matrix Having relation Between Files and developers



```
00000000100000000000000000000100000000000000000000000000000000000000000000000000000000000000000000000000001
00100000100000000001000000010000000000000000000000000000000100000000000000000001000000000000000000000000001
01101001001001010000100000001100001110001000000000100011011110000000010100000000000101000100000011000100001
00000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000001
00001000000000010000000000100000000000000000000000000000000000000000000000000000000000000000000000000000001
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000000000000000000
00000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
00100000100100000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000
00000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
00000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000001
00000000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000
00000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000101000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000
```

CONTAINS

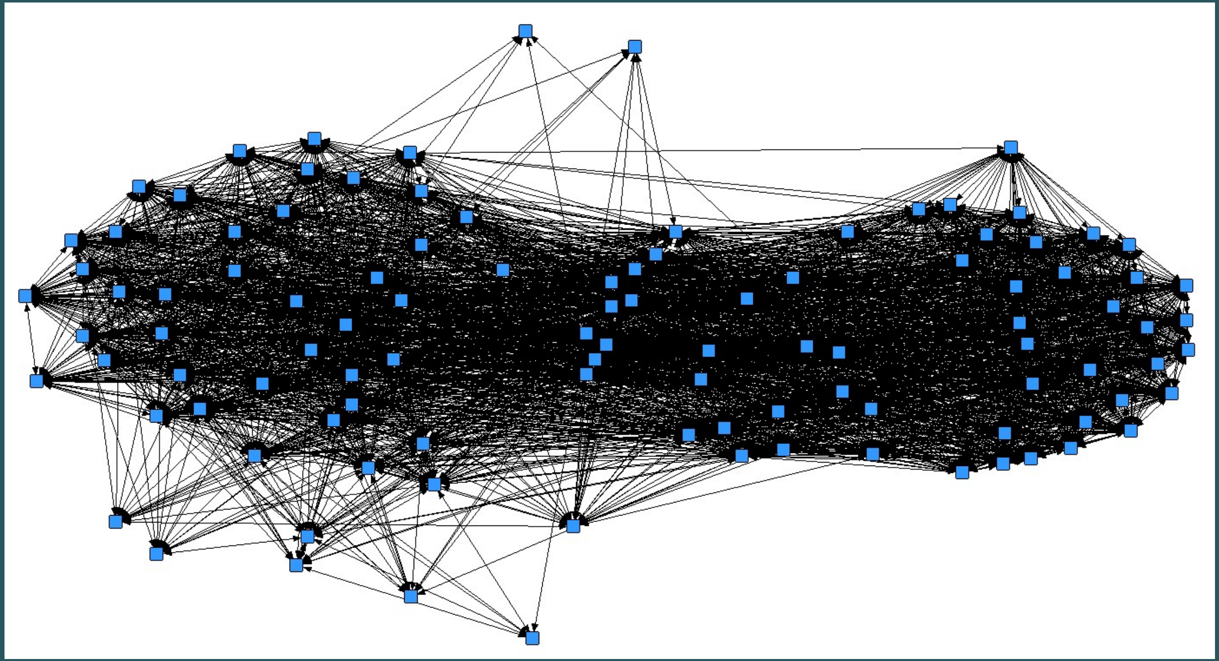
- ✓ 106 COLUMNS WHICH REPRESENT TOP DEVELOPERS

AND

- ✓ 12579 ROWS WHICH REPRESENT EDITED FILES BY TOP DEVELOPERS.

Socio-technical network analysis

- ✓ We used UCINET 6
- ✓ Two mode network analysis feature
- ✓ Convert two mode to one mode data By columns (developers)



Thanks!

No question