

# Appendix B: Cluster Identification and Transition Analysis in Unicage

This code replicates the functionality found in appendix A one step at a time, using shell programming and the Unicage development platform. Each of the scripts listed here is found at <https://github.com/Removed for Anonymity>

## Running these scripts

This analysis process is separated to 7 steps. You can run each or all steps using the helper script

`twitter_analysis.sh` as follows:

```
$ twitter_analysis.sh <start_step_no> <end_step_no>
```

A key to the step numbers is:

- 1 - `list_word_pairings.sh`
- 2 - `wgted_edge_gen.sh`
- 3 - `unwgted_edge_gen.sh`
- 4 - `run_mcliques.sh`
- 5 - `run_cos.sh`
- 6 - `back_to_org_words.sh`
- 7 - `compute_transition_likelihoods.sh`

For example, to execute step4 to step6:

```
shell
$ twitter_analysis.sh 4 6
```

To execute step2 only:

```
shell
$ twitter_analysis.sh 2 2
```

To execute all steps:

```
shell
$ twitter_analysis.sh 1 7
```

## Step 1: `list_word_pairings.sh`

This script creates lists of hashtag pairs from json files.

Output: produces `DATA/result.XXXX.`

```
#!/bin/bash

homed=/home/James.P.H/UNICAGE
toold=${homed}/TOOL
shelld=${homed}/SHELL
rawd=/home/James.P.H/data
semd=${homed}/SEMAPHORE
datad=${homed}/DATA
workd=${homed}/twitter

tmp=/tmp/$$

mkdir -p ${datad}

n=0

# Process zipped files/
echo ${rawd}/posts_sample*.gz |
tarr |
while read zipfile; do
    n=$((n+1))
    echo $zipfile $n

    {
        zcat $zipfile |
        ${homed}/SHELL/myjsonparser |
        # 1: "time" 2: timestamp (epoch msec) 3: "hashtag" 4-N: hashtags

        awk 'NF>5{for(i=4;i<=NF;i++)for(j=i+1;j<=NF;j++){print $i,$j,int($2/1000)}}' |
        # list all possible 2 word combinations with timestamp. 1: word1 2: word2 3: timestamp (epoch msec)

        TZ=UTC calclock -r 3 |
        # 1: word1 2: word2 3: timestamp (epoch sec) 4: timestamp (YYYYMMDDhhmmss)

        self 1 2 4.1.8 |
        # 1: word1 2: word2 3: timestamp (YYYYMMDD)

        msort key=1/3 |
```

```

count 1 3 > ${datad}/result.$n
# count lines having the same word combination and timestamp 1:word1 2:word2 3:date 4:count

# run 5 processes in parallel
touch ${semd}/sem.$n
} &
if [ $((n % 5)) -eq 0 ]; then
    eval semwait ${semd}/sem.${$((n-4))}..$n
    eval rm -f ${semd}/sem.* 2> /dev/null
fi
done

wait

n=$(ls ${datad}/result.* | sed -e 's/\./ /g' | self NF | msort key=1n | tail -1)

# Process unzipped files.
# *There are unzipped files in raw data dir(/home/James.P.H/data).
echo ${rawd}/posts_sample* |
tarr |
self 1 1.-3.3 |
delr 2 '.gz' |
self 1 |
while read nozipfile; do
    n=$((n+1))
    echo $nozipfile $n

    {
        cat $nozipfile |
        ${homed}/SHELL/myjsonparser |
        # 1: "time" 2: timestamp (epoch msec) 3: "hashtag" 4-N: hashtags

        awk 'NF>5{for(i=4;i<=NF;i++)for(j=i+1;j<=NF;j++){print $i,$j,int($2/1000)}}' |
        # list all possible 2 word combinations with timestamp. 1: word1 2: word2 3: timestamp (epoch msec)

        calclock -r 3 |
        # 1: word1 2: word2 3: timestamp (epoch sec) 4: timestamp (YYYYMMDDhhmmss)

        self 1 2 4.1.8 |
        # 1: word1 2: word2 3: timestamp (YYYYMMDD)

        msort key=1/3 |
        count 1 3 > ${datad}/result.$n
        # count lines having the same word combination and timestamp 1:word1 2:word2 3:date 4:count
    }
done

```

```

# run 5 processes in parallel
touch ${semd}/sem.$n
} &
if [ $((n % 5)) -eq 0 ]; then
    eval semwait ${semd}/sem.${$((n-4))}..$n
    eval rm -f ${semd}/sem.* 2> /dev/null
fi
done

#semwait "${semd}/sem.*"
wait
eval rm -f ${semd}/sem.* 2> /dev/null

rm -f $tmp-*

exit 0

```

## Step 2: `wgted_edge_gen.sh`

This script creates weighted edgelists from `result.*` and places them under `yyyyymmdd` dirs.

Output: produces `twitter/yyyyymmdd/weighted_edges_yyyyymmdd.txt`

```

#!/bin/bash -xv

# wgted_edge_gen.sh creates weighted edgelists from result.*
# and place them under yyyyymmdd dirs.

homed=/home/James.P.H/UNICAGE
toold=${homed}/TOOL
shelld=${homed}/SHELL
rawd=/home/James.P.H/data
semd=${homed}/SEMAPHORE
datad=${homed}/DATA
workd=${homed}/twitter

# TODO debug
#datad=${homed}/DATA.mini
#workd=${homed}/twitter.mini

tmp=/tmp/$$

```

```

# error function: show ERROR and exit with 1
ERROR_EXIT() {
    echo "ERROR"
    exit 1
}

mkdir -p ${workd}

# count the number of files
n=$(ls ${datad}/result.* | gyo)

for i in $(seq 1 ${n} | tarr)
do
    # 1:Tag1 2:Tag2 3:date 4:count
    sorter -d ${tmp}-weighted_edges_%3_${i} ${datad}/result.${i}
    [ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT
done

# listup target dates
echo ${tmp}-weighted_edges_??????_?_* |
tarr |
ugrep -v '\?' |
sed -e 's/_/ /g' |
self NF-1 |
msort key=1 |
uniq > ${tmp}-datelist
# 1:date(YYYYMMDD)

[ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

for day in $(cat ${tmp}-datelist); do
    mkdir -p ${workd}/${day}

    cat ${tmp}-weighted_edges_${day}_* |
    # 1:word1 2:word2 3:count
    msort key=1/2 |
    sm2 1 2 3 3 > ${workd}/${day}/weighted_edges_${day}.txt
    # 1:word1 2:word2 3:count

    [ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

done

rm ${tmp}-*

```

```
exit 0
```

## Step 3: `unwgted_edge_gen.sh`

This script creates unweighted edgelists under the same dir sorted by threshold dirs.

Output: produces `twitter/yyyymmdd/th_XX/unweightedyyyymmdd_th_XX.txt`

```
#!/bin/bash -xv

# unwgted_edge_gen.sh expects weighted edgelists
# (weighted_edges/yyyymmdd.txt) located in
# /home/James.P.H/UNICAGE/twitter/yyyymmdd
# and creates unweighted edgelists under the same dir
# sorted by threshold dirs.

homed=/home/James.P.H/UNICAGE
toold=${homed}/TOOL
shelld=${homed}/SHELL
rawd=/home/James.P.H/data
semd=${homed}/SEMAPHORE
datad=${homed}/DATA
workd=${homed}/twitter

# TODO test
#datad=${homed}/DATA.mini
#workd=${homed}/twitter.mini

tmp=/tmp/$$

# error function: show ERROR and delete tmp files
ERROR_EXIT() {
    echo "ERROR"
    rm -f $tmp-*
    exit 1
}

# setting threshold
seq 2 15 | maezero 1.2 > $tmp-threshold
[ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

# creating header file
itouch "Hashtag1 Hashtag2 count" $tmp-header
```

```

[ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

# create list for all pairs of thresholds and filenames
echo ${workd}/201[45]*/weighted_edges_*.txt      |
tarr                                              |
joinx $tmp-threshold -                          |
# 1:threshold 2:filename
while read th wgtedges ; do
    echo ${wgtedges}
    [ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

    # define year-month-date variable for dir and file name
    yyyymmdd=$(echo ${wgtedges} | awk -F \/ '{print $(NF-1)}')
    [ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

    echo ${yyyymmdd} th_${th}
    [ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

    # create threshold dirs under twitter/YYYYMMDD
    mkdir -p $(dirname ${wgtedges})/th_${th}
    [ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

    cat $tmp-header ${wgtedges}                  |
    # output lines whose count feild is above thresholds
    ${toold}/tagcond '%count > "${th}"'          |
    # remove threshold feild
    tagself Hashtag1 Hashtag2                    |
    # remove header
    tail -n +2 > ${workd}/${yyyymmdd}/th_${th}/unweighted_${yyyymmdd}_th_${th}.txt
    [ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

done
[ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

# delete tmp files
rm -f $tmp-*

exit 0

```

## Step 4: `run_mcliques.sh`

This script executes maximal\_cliques to all unweigthed edges.

Output: produces

- `twitter/yyyymmdd/th_XX/unweighted_edges_yyyyyymmdd_th_XX.txt.map`
- `twitter/yyyymmdd/th_XX/unweighted_edges_yyyyyymmdd_th_XX.txt.mcliques`

```
#!/bin/bash -xv

# run_mcliques.sh executes maximal_cliques to all unweigthed edges.
# produce unweighted_edges_yyyyyymmdd.txt.map and unweighted_edges_yyyyyymmdd.txt.mcliques

homed=/home/James.P.H/UNICAGE
toold=${homed}/TOOL
shelld=${homed}/SHELL
rawd=/home/James.P.H/data
semd=${homed}/SEMAPHORE
datad=${homed}/DATA
workd=${homed}/twitter

# TODO test
#datad=${homed}/DATA.mini
#workd=${homed}/twitter.mini

# error function: show ERROR
ERROR_EXIT() {
    echo "ERROR"
    exit 1
}

# 共有ライブラリヘパスを通す(maximal_cliques用)
LD_LIBRARY_PATH=/usr/local/lib:/usr/lib
export LD_LIBRARY_PATH

# running maximal_cliques
for unwgtd_edges in ${workd}/*/th_*/unweighted_*_th_*.txt
do
    echo "Processing ${unwgtd_edges}."
    [ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT

    # skip empty files
    if [ ! -s ${unwgtd_edges} ] ; then
        echo "Skipped $(basename ${unwgtd_edges})."
        continue
    fi
done
```



```
cd $(dirname ${unwgted_edges})
[ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT

${toold}/maximal_cliques ${unwgted_edges}
[ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT
# unweighted_edges_YYYYMMDD.txt.map (1:Tag 2:integer)
# unweighted_edges_YYYYMMDD.txt.mcliques (1...N: integer for nodes N+1: virtual node -1)

echo "${unwgted_edges} done."
[ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT
done

exit 0
```

## Step 5: `run_cos.sh`

---

This script executes `cos` using `*.mcliques` files to create communities.

Output: produces `twitter/YYYYMMDD/th_XX/N_communities.txt`

```

#!/bin/bash -xv

# run_cos.sh creates communities using *.mcliques files.

homed=/home/James.P.H/UNICAGE
toold=${homed}/TOOL
shelld=${homed}/SHELL
rawd=/home/James.P.H/data
semd=${homed}/SEMAPHORE
datad=${homed}/DATA
workd=${homed}/twitter

# error function: show ERROR
ERROR_EXIT() {
echo "ERROR"
exit 1
}

# 共有ライブラリヘパスを通す(cos用)
LD_LIBRARY_PATH=/usr/local/lib:/usr/lib
export LD_LIBRARY_PATH

# running cos
for mcliques in ${workd}/*/th_*/unweighted_*_th_*.txt.mcliques
do
    echo "Processing ${mcliques}."
    [ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT

    # changing dir so that output files can be saved under each th dirs.
    cd $(dirname ${mcliques})
    [ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT

    ${toold}/cos ${mcliques}
    [ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT
    # N_communities.txt (1:community_id 2..N: maximal_clique)
    # k_num_communities.txt (1:k 2: number of k-clique communities discovered)

    echo "${mcliques} done."
    [ $(plus $(echo ${PIPESTATUS[@]})) -eq "0" ] || ERROR_EXIT
done

exit 0

```

## Step 6: `back_to_org_words.sh`

This script reverts integers in `N_communities.txt` to original words using map file generated by `maximal_cliques`.

Output: produces `twitter/yyyymmdd/th_XX/namedN_communities.txt`

```
#!/bin/bash -xv

# back_to_org_words.sh:
# use map file generated by maximal_cliques to revert integers in N_communities.txt
# to original words.

homed=/home/James.P.H/UNICAGE
toold=${homed}/TOOL
shelld=${homed}/SHELL
rawd=/home/James.P.H/data
semd=${homed}/SEMAPHORE
datad=${homed}/DATA
workd=${homed}/twitter

tmp=/tmp/$$

# TODO test
#datad=${homed}/DATA.mini
#workd=${homed}/twitter.mini

# error function: show ERROR and delete tmp files
ERROR_EXIT() {
    echo "ERROR"
    rm -f $tmp-*
    exit 1
}

echo ${workd}/*/*th_*/*[0-9]*_communities.txt |
tarr |
ugrep -v '\*' |

# community番号とthreshold数を変数に入れてwhileループをする必要がある

while read community_files; do
```

```

:>$tmp-tran

echo ${community_files}

# get directory path of target-file
dirname=$(dirname ${community_files})
# get filename
filename=$(basename ${community_files})

# read a community file
fused 's:// /1' ${community_files} |
# 1: community id 2..N: integer for node

# remove unnecessary space char at the end of each line
sed -e 's/ *$//' |

# remove lines which have only 1 field for community id
gawk 'NF>1' |

tarr num=1 |
# 1: community id 2: integer

self 2 1 |
# 1: integer 2: community id
# sort by field 1/2
msort key=1/2 |
# remove the same records, only take last one
getlast 1 2 > $tmp-tran
# 1: integer 2: community id

[ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

# TODO: for debug
cat $tmp-tran

# Read the word-map file
cat ${dirname}/unweighted_*_th_*.txt.map |
# 1: word 2: integer
self 2 1 |
# 1: integer 2: word
# sort by field 1
msort key=1 |
# join map file to community -tran
join1 key=1 - $tmp-tran |
# 1: integer 2: word 3: community id

```

```

self 3 2 |
# 1: community id 2: word
yarr num=1 > ${dirname}/named${filename}
# 1: community id 2..N: word1..N

[ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

done
[ $(plus $(echo "${PIPESTATUS[@]}")) -eq "0" ] || ERROR_EXIT

# delete tmp files
rm -f $tmp-*

exit 0

```

## Step 7: `compute_transition_likelihoods.sh`

This script will compute transition-likelihoods map files using `named_N_communities.txt`.

Output: produces `twitter/yyyyymmdd/th_XX/namedN_communities_transition.csv`

```

#!/bin/bash -xv

# compute_transition_likelihoods.sh
#

homed=/home/James.P.H/UNICAGE
toold=${homed}/TOOL
shelld=${homed}/SHELL
rawd=/home/James.P.H/data
semd=${homed}/SEMAPHORE
datad=${homed}/DATA
workd=${homed}/twitter

tmp=/tmp/$$

# TODO test
#shelld=${homed}/SHELL/sugi_test
#datad=${homed}/DATA.mini
#workd=${homed}/twitter.mini

# error function: show ERROR and delete tmp files

```

```

ERROR_EXIT() {
    echo "ERROR"
    rm -f $tmp-*
    exit 1
}

# 対象の日付リストを作成
echo ${workd}/2*          |
tarr                      |
ugrep -v '\*'            |
sed -e 's/\\/ /g'         |
self NF                  |
msort key=1               > $tmp-date-dir-list
# 1:date(real dir)

[ $(plus $(echo "${PIPESTATUS[@]}") -eq "0" ) || ERROR_EXIT

fromdate=$(head -1 $tmp-date-dir-list)
todate=$(tail -1 $tmp-date-dir-list)

mdate -e ${fromdate} ${todate}          > $tmp-date-list
# 1:date

echo ${workd}/20??????/th*/named*_communities.txt |
tarr                                              |
grep -v '\*'                                    |
# 1:current_filename
self 1 1                                         |
fsed 's#/# #2'                                  |
# 1:current_filename ... NF-2:current_date NF-1:"th_XX" NF:"namedXX_communities.txt"
self 1 NF-2/NF                                  |
# 1:current_filename 2:current_date 3:"th_XX" 4:"namedXX_communities.txt"
mdate -f 2/+1                                   |
# 1:current_filename 2:current_date 3:next_date 4:"th_XX" 5:"namedXX_communities.txt"
gawk '{ print $1, "'${workd}'/"$3/"$4/"$5 }'      |
# 1:current_filename 2:next_filename
${shelld}/intersection
## output result to "${workd}/named*_communities_transition.work" each line.

# create map file
echo ${workd}/20??????/th*/named*_communities_transition.work |
tarr                                              |
grep -v '\*'                                    |
while read filename; do
    # csv file name

```

```
    csv_filename=$(echo $filename | sed 's/\./ /g' | self 1/NF-1 | sed 's/ /./g' | gawk '{ p

# create map: index=id(curr) columns=id(next)
maezero 1.3 2.3 $filename      |
map num=1x1 -                  |
# convert to csv
tocsv                          > ${csv_filename}
done

# delete tmp files
rm ${workd}/20??????/th*/named*_communities_transition.work
rm -f $tmp-*

exit 0
```