**RecSys vsu’s source code**

By Vladimir Nikulin

*Vyatka State University, Russia*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Copyright [2013] [vsu Team]**

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS"

BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,

either express or implied. See the License for the specific language governing permissions and limitations under the License.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**SOURCE CODE:**

#\*\*\* written by Vladimir Nikulin, Vyatka State University (vsu)

#\*\*\* DATE: 20-09-2013

#\*\*\* Kaggle/RecSys2013

#\*\*\* all directories MUST be adjusted properly

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#\*\*\* SOME IMPORTANT NOTATIONS:

# YI – Yelp Index (textual)

# nrw – number of reviews

# seqInd – sequential index (sequential number)

# prStar – predicted Star

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

R-code to extract train\_user.txt from given JSON file

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

library(rjson)

# Convert raw JSON file into managable data frame

UnpackJSON <- function(filePath){

con <- file(filePath, "r")

input <- readLines(con)

jsonData <- sapply(input,fromJSON)

close(con)

df <- data.frame(jsonData)

temp <- rownames(df)

df <- as.data.frame(t(df))

colnames(df) <- temp

rownames(df) <- NULL

return(df)

}

# Convert the nested lists into regular vectors

UnlistJSON <- function(df){

for(i in 1:ncol(df)) {

temp <- unlist(df[,i])

names(temp) <- NULL

df[,i] <- temp

}

return(df)

}

# Convert the votes column from a list into 3 seperate columns for useful, funny, cool

UnwrapVotes <- function(df){

temp <- unlist(df$votes)

names(temp) <- NULL

index <- seq(from=1, to=length(temp)-2, by=3)

df$funny <- temp[index]

index <- seq(from=2, to=length(temp)-1, by=3)

df$useful <- temp[index]

index <- seq(from=3, to=length(temp), by=3)

df$cool <- temp[index]

return(df)

}

# Training User data

filePath <- "c:/contest/Kaggle/RecSys/yelp\_training\_set/yelp\_training\_set\_user.json"

user <- UnpackJSON(filePath)

user <- UnwrapVotes(user)

user <- user[,c('user\_id', 'average\_stars', 'review\_count', 'useful', 'funny', 'cool')]

user <- UnlistJSON(user)

write.table(user, file = "c:/contest/Kaggle/RecSys/output/train\_user.txt", quote=FALSE, row.names=FALSE, col.names=TRUE, sep = " ")

########################################################

R-code to extract train\_business.txt from given JSON file

library(rjson)

# Convert raw JSON file into managable data frame

UnpackJSON <- function(filePath){

con <- file(filePath, "r")

input <- readLines(con)

jsonData <- sapply(input,fromJSON)

close(con)

df <- data.frame(jsonData)

temp <- rownames(df)

df <- as.data.frame(t(df))

colnames(df) <- temp

rownames(df) <- NULL

return(df)

}

# Convert the nested lists into regular vectors

UnlistJSON <- function(df){

for(i in 1:ncol(df)) {

temp <- unlist(df[,i])

names(temp) <- NULL

df[,i] <- temp

}

return(df)

}

# Training User data

filePath <- "c:/contest/Kaggle/RecSys/yelp\_training\_set/yelp\_training\_set\_business.json"

user <- UnpackJSON(filePath)

user <- user[,c('business\_id', 'name', 'longitude', 'stars', 'latitude', 'review\_count', 'city')]

user <- UnlistJSON(user)

write.table(user, file = "c:/contest/Kaggle/RecSys/output/train\_business.txt", quote=FALSE, row.names=FALSE, col.names=TRUE, sep = "\t")

########################################################

library(rjson)

# Convert raw JSON file into managable data frame

UnpackJSON <- function(filePath){

con <- file(filePath, "r")

input <- readLines(con)

jsonData <- sapply(input,fromJSON)

close(con)

df <- data.frame(jsonData)

temp <- rownames(df)

df <- as.data.frame(t(df))

colnames(df) <- temp

rownames(df) <- NULL

return(df)

}

# Convert the nested lists into regular vectors

UnlistJSON <- function(df){

for(i in 1:ncol(df)) {

temp <- unlist(df[,i])

names(temp) <- NULL

df[,i] <- temp

}

return(df)

}

# Training User data

filePath <- "c:/contest/Kaggle/RecSys/final\_data/final\_test\_set\_business.json"

user <- UnpackJSON(filePath)

user <- user[,c('business\_id', 'name', 'longitude', 'latitude', 'review\_count', 'city')]

user <- UnlistJSON(user)

write.table(user, file = "c:/contest/Kaggle/RecSys/output\_final/test\_bus\_d230813.txt", quote=FALSE, row.names=FALSE, col.names=TRUE, sep = "\t")

########################################################

R-code to extract tst\_review\_d230813.txt from given JSON file

library(rjson)

# Convert raw JSON file into managable data frame

UnpackJSON <- function(filePath){

con <- file(filePath, "r")

input <- readLines(con)

jsonData <- sapply(input,fromJSON)

close(con)

df <- data.frame(jsonData)

temp <- rownames(df)

df <- as.data.frame(t(df))

colnames(df) <- temp

rownames(df) <- NULL

return(df)

}

# Convert the nested lists into regular vectors

UnlistJSON <- function(df){

for(i in 1:ncol(df)) {

temp <- unlist(df[,i])

names(temp) <- NULL

df[,i] <- temp

}

return(df)

}

# Training User data

filePath <- "c:/contest/Kaggle/RecSys/final\_data/final\_test\_set\_review.json"

user <- UnpackJSON(filePath)

user <- user[,c('user\_id', 'business\_id')]

user <- UnlistJSON(user)

write.table(user, file = "c:/contest/Kaggle/RecSys/output\_final/tst\_review\_d230813.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

########################################################

#!/usr/local/bin/perl

#\*\*\*\* 31st August 2013 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### **Perl/LineN1.pl**  #######################

#\*\*\* main advantage of Perl: work well with Yelp textual indexes

#=============================================

#\*\*\* INPUT: ff) YI seqInd nrw (separated by the empty space);

# fa) YI seqInd nrw; fb) see above (empty space separated); fc) see above (TAB separated);

# fd) YI prStar nrw; fe) YI prStar nrw; fx) above (empty space separated).

#\*\*\* OUTPUT: f1) Line N1 solution; f2) statistics.

#\*\*\* Remark. Method calculation of prStars for both bus and users is given in Appendix N2 (page 30).

#=============================================

open(ff, "c:/contest/Kaggle/RecSys/data\_final/library\_user.txt");

open(fa, "c:/contest/Kaggle/RecSys/data\_final/library\_business.txt");

open(fb, "c:/contest/Kaggle/RecSys/output/train\_user.txt");

open(fc, "c:/contest/Kaggle/RecSys/output/train\_business.txt");

open(fd, "c:/contest/Kaggle/RecSys/output\_final/p3/star\_count\_bus\_d280813\_tst.txt");

open(fe, "c:/contest/Kaggle/RecSys/output\_final/p3/star\_count\_user\_d280813\_tst.txt");

open(fx, "c:/contest/Kaggle/RecSys/output\_final/tst\_review\_d230813.txt");

#===========================================

open(f1, ">c:/contest/Kaggle/RecSys/solution\_final/p6/tst\_vsu\_review\_d310813b.txt");

open(f2, ">c:/contest/Kaggle/RecSys/solution\_final/p6/stats\_vsu\_d310813b.txt");

#===========================================

while(defined($\_ = <ff>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

$su{$a}{0} = $b;

for($i=1;$i<=2;$i++){

$su{$a}{$i} = 0;

}

}

#===========================================

while(defined($\_ = <fa>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

$sb{$a}{0} = $b;

for($i=1;$i<=2;$i++){

$sb{$a}{$i} = 0;

}

}

#===========================================

$\_ = <fb>; #\*\*\* read title

while(defined($\_ = <fb>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

$c = @fields[2];

$su{$a}{0} = 0;

$su{$a}{1} = $b;

$su{$a}{2} = $c;

}

#===========================================

$\_ = <fc>; #\*\*\* read title

while(defined($\_ = <fc>)){

chomp $\_;

@fields = split(/\t/,$\_);

$a = @fields[0];

$b = @fields[3];

$c = @fields[5];

$sb{$a}{0} = 0;

$sb{$a}{1} = $b;

$sb{$a}{2} = $c;

}

#===========================================

while(defined($\_ = <fd>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

$c = @fields[2];

$zb{$a}{0} = 0;

$zb{$a}{1} = $b;

$zb{$a}{2} = $c;

}

#===========================================

while(defined($\_ = <fe>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

$c = @fields[2];

$zu{$a}{0} = 0;

$zu{$a}{1} = $b;

$zu{$a}{2} = $c;

}

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$alfa = 5.0;

$b0 = 0.05;

$b1 = 0.15;

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

$i1 = 0;$i11 = 0;

$i2 = 0;$i21 = 0;

$i3 = 0;$i31 = 0;

$i4 = 0;$i41 = 0;

while(defined($\_ = <fx>)){

chomp $\_;

@fields = split(/ /,$\_);

$au = @fields[0];

$ab = @fields[1];

$ind = 0;

$ncf = 0;

if($su{$au}{1}>=1){

if(($sb{$ab}{1}>=1) && ($sb{$ab}{2}>=1) && ($su{$au}{2}>=1)){

$xu = $su{$au}{2};

$xb = $sb{$ab}{2};

$psi = $su{$au}{2}/20.0;

if($psi > 1.0){$psi = 1.0;}

$beta = $psi\*$b0 + (1.0-$psi)\*$b1;

if(defined($eu{$au})){$ru = ($su{$au}{1} + $beta\*$eu{$au})/(1.0+$beta);}

else{$ru = $su{$au}{1};}

$psi = $sb{$ab}{2}/50.0;

if($psi > 1.0){$psi = 1.0;}

$beta = $psi\*$b0 + (1.0-$psi)\*$b1;

if(defined($eb{$ab})){$rb = ($sb{$ab}{1} + $beta\*$eb{$ab})/(1.0+$beta);}

else{$rb = $sb{$ab}{1};}

$xs = $xu + $xb;

$q1 = ($xu\*log($ru) + $xb\*log($rb))/$xs;

$q = exp($q1);

$ind = 1;

$ncf = $xs/2.0;

$i1++;

}

else{

$psi = $su{$au}{2}/20.0;

if($psi > 1.0){$psi = 1.0;}

$beta = $psi\*$b0 + (1.0-$psi)\*$b1;

if(defined($eu{$au})){$ru = ($su{$au}{1} + $beta\*$eu{$au})/(1.0+$beta);}

else{$ru = $su{$au}{1};}

$q = $ru;

$ind = 2;

$ncf = $su{$au}{2};

$i2++;

}

}

elsif($sb{$ab}{1}>=1){

$psi = $sb{$ab}{2}/50.0;

if($psi > 1.0){$psi = 1.0;}

$beta = $psi\*$b0 + (1.0-$psi)\*$b1;

if(defined($eb{$ab})){$rb = ($sb{$ab}{1} + $beta\*$eb{$ab})/(1.0+$beta);}

else{$rb = $sb{$ab}{1};}

$q = $rb;

$ind = 3;

$ncf = $sb{$ab}{2};

$i3++;

}

else{

if($zb{$ab}{1}>=1){

if(($zu{$au}{1}>=1) && ($zu{$au}{2}>=1) && ($zb{$ab}{2}>=1)){

$xu = $zu{$au}{2};

$xb = $alfa\*$zb{$ab}{2};

$xs = $xu + $xb;

$q1 = ($xu\*log($zu{$au}{1}) + $xb\*log($zb{$ab}{1}))/$xs;

$q = exp($q1);

$ind = 4;

$ncf = $xs/2.0;

$i11++;

}

else{

$q = $zb{$ab}{1};

$ind = 5;

$ncf = $zb{$ab}{2};

$i21++;

}

}

elsif($zu{$au}{1}>=1){

$q = $zu{$au}{1};

$ind = 6;

$ncf = $zu{$au}{2};

$i31++;

}

else{

$q = 0.0;

$ind = 7;

$ncf = 0;

$i41++;

}

}

print f1 "$q $ncf $ind\n";

}

print f2 "$i1 $i2 $i3 $i4\n";

print f2 "$i11 $i21 $i31 $i41\n";

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

close(ff);

close(fa);

close(fb);

close(fc);

close(fd);

close(fe);

close(fx);

close(f1);

close(f2);

close(f3);

close(fs);

########################################################

#!/usr/local/bin/perl

### Line N3: preparation of the train-data #####################

#===========================================

open(ff, "c:/contest/Kaggle/RecSys/rdata\_final/p2/trn\_user\_d270813.txt");

open(fa, "c:/contest/Kaggle/RecSys/rdata\_final/p2/trn\_bus\_d270813.txt");

open(fb, "c:/contest/Kaggle/RecSys/data\_final/trn\_review\_d240813.txt");

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

open(fu, ">c:/contest/Kaggle/RecSys/rdata\_final/p2/trn\_review\_d270813.txt");

open(fs, ">c:/contest/Kaggle/RecSys/rdata\_final/p2/stats\_review\_d270813\_trn.txt");

#===========================================

$ii = 0;

while(defined($\_ = <ff>)){

chomp $\_;

$ii++;

$su{$ii} = $\_;

}

#===========================================

$ii = 0;

while(defined($\_ = <fa>)){

chomp $\_;

$ii++;

$sb{$ii} = $\_;

}

#===========================================

print fu "tg ";

for($i=1;$i<=58;$i++){

print fu "v$i ";

}

$i = 59;

print fu "v$i\n";

#===========================================

$ii = 0;

while(defined($\_ = <fb>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

$c = @fields[2];

if(defined($su{$a}) && defined($sb{$b})){

$q = $su{$a};

$x = $sb{$b};

print fu "$c $q $x\n";

}

else{$ii++;}

}

#===========================================

print fs "$ii\n";

#===========================================

close(ff);

close(fa);

close(fb);

close(fu);

close(fs);

########################################################

#!/usr/local/bin/perl

### Line N3: preparation of the test-data ###############

#===========================================

open(ff, "c:/contest/Kaggle/RecSys/rdata\_final/p2/trn\_user\_d270813.txt");

open(fa, "c:/contest/Kaggle/RecSys/rdata\_final/p2/trn\_bus\_d270813.txt");

open(fb, "c:/contest/Kaggle/RecSys/data\_final/tst\_review\_d240813.txt");

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

open(fu, ">c:/contest/Kaggle/RecSys/rdata\_final/p2/tst\_review\_d270813.txt");

open(fs, ">c:/contest/Kaggle/RecSys/rdata\_final/p2/stats\_review\_d270813\_tst.txt");

#===========================================

$ii = 0;

while(defined($\_ = <ff>)){

chomp $\_;

$ii++;

$su{$ii} = $\_;

}

#===========================================

$ii = 0;

while(defined($\_ = <fa>)){

chomp $\_;

$ii++;

$sb{$ii} = $\_;

}

#===========================================

for($i=1;$i<=58;$i++){

print fu "v$i ";

}

$i = 59;

print fu "v$i\n";

#===========================================

$ii = 0;

while(defined($\_ = <fb>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

if(defined($su{$a}) && defined($sb{$b})){

$q = $su{$a};

$x = $sb{$b};

print fu "$q $x\n";

}

else{$ii++;}

}

#===========================================

print fs "$ii\n";

#===========================================

close(ff);

close(fa);

close(fb);

close(fu);

close(fs);

########################################################

Computation of the Line N3 solution: gbm\_vsu\_review\_d270813.txt

trace\_vsu\_gbm\_review\_d270813.txt : trace with CV-passports (homogeneous ensembling)

library(gbm)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

A <- read.table("/home/user623/RecSys/rdata\_final/trn\_review\_d270813.txt", header=TRUE)

T <- read.table("/home/user623/RecSys/rdata\_final/tst\_review\_d270813.txt", header=TRUE)

##################################################

function(n){

alfa = 0.15

m = 50

trace = matrix(0,m+1,4)

nr = nrow(A)

nt = nrow(T)

sol = matrix(0,nt,1)

solut = matrix(0,nr,2)

ind = matrix(0,nr,1)

print("start")

for(jj in 1:m){

for(i in 1:nr){

q = runif(1)

if(q<=alfa){ind[i,1] = 1}

else{ind[i,1] = 0}

}

object <- gbm(tg~.,data=A[ind==1,],distribution="gaussian",n.trees=2000,shrinkage=0.01,interaction.depth=16,n.minobsinnode=5)

sol = sol + predict(object,newdata=T,n.trees=2000,type="response")

tr = predict(object,newdata=A,n.trees=2000,type="response")

S = 0.0

k = 0

S1 = 0.0

k1 = 0

for(i in 1:nr){

if(tr[i]<1.0){tr[i] = 1.0}

else if(tr[i]>5.0){tr[i] = 5.0}

if(ind[i,1] == 1){

S1 = S1 + (A[i,1] - tr[i])^2.0

k1 = k1 + 1

}

else{

solut[i,1] = solut[i,1] + 1

solut[i,2] = solut[i,2] + tr[i]

S = S + (A[i,1] - tr[i])^2.0

k = k + 1

}

}

trace[jj,1] = jj

trace[jj,2] = k/nr

trace[jj,3] = sqrt(S/k)

trace[jj,4] = sqrt(S1/k1)

print(trace[jj,])

}

S = 0.0

i1 = 0

for(i in 1:nr){

if(solut[i,1]>=1){

i1 = i1+1

if(solut[i,1] >= 2){

solut[i,2] = solut[i,2]/solut[i,1]

}

if(solut[i,2]<1.0){solut[i,2] = 1.0}

else if(solut[i,2]>5.0){solut[i,2] = 5.0}

S = S + (A[i,1] - solut[i,2])^2.0

}

}

trace[m+1,1] = jj

trace[m+1,2] = i1

trace[m+1,3] = sqrt(S/i1)

print(trace[m+1,])

sol <- sol/m

write.table(trace, file = "/home/user623/RecSys/output\_final/p2/trace\_vsu\_gbm\_review\_d270813.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

write.table(sol, file = "/home/user623/RecSys/output\_final/p2/gbm\_vsu\_review\_d270813.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

}

########################################################

//\*\*\*\* heterogeneous ensemble of the final solution \*\*\*\*\*\*\*\*\*\*\*\*\*

// ff) seqInd\_user seqInd\_bus Star; fa) seqInd\_user seqInd\_bus; fi) see Appendix (page 27).

//#######################################################

#include<io.h>

#include<math.h>

#include<stdlib.h>

#include<stdio.h>

#include <time.h>

#include <fstream>

#include "MersenneTwister.h"

void showrate( clock\_t start, clock\_t stop, int reps );

int main(void)

{

int i,i1,i2,i3,ind,ind1,j,k,k1,m,m1,n,hp[9];

int g[99][3],f[99][3];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int nr=229907; //\*\*\* training

const int nt=36404; //\*\*\* testing

const int nu=55965;

const int ni=14334;

const int alfa=89;

const float gamma=0.25;

const float beta=0.2;

const float phi\_gmf=0.02;

const float tau=2.0;

const float delta=0.5;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float vs,p1,p2,q,q1,q2,r1,r2,pru[99],prb[99],frac\_user,est\_user;

float S,frac\_bus,est\_bus,q\_cnt,q\_count,solut;

//================================================

int \*A,\*u,\*v;

//================================================

FILE \*ff,\*fa,\*fb,\*fi,\*fc,\*f1;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ff=fopen("c:/contest/Kaggle/RecSys/data\_final/trn\_review\_num.txt","r");

fa=fopen("c:/contest/Kaggle/RecSys/data\_final/tst\_review\_num.txt","r");

fb=fopen("c:/contest/Kaggle/RecSys/solution\_final/p6/tst\_vsu\_review\_d310813b.txt","r"); //\*\*\* Line N1

fi=fopen("c:/contest/Kaggle/RecSys/data\_final/tst\_review\_d240813.txt","r");

fc=fopen("c:/contest/Kaggle/RecSys/output\_final/ p2/gbm\_vsu\_review\_d270813.txt","r"); //\*\*\* Line N3

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

f1=fopen("c:/contest/Kaggle/RecSys/solution\_final/p6/tst\_vsu\_stars\_d310813b.txt","w"); //\*\*\* final solution

//++++++++++++++++++++++++++++++++++++++++++++++++

A = (int \*)calloc(8000001, sizeof(int));

u = (int \*)calloc(900001, sizeof(int));

v = (int \*)calloc(200001, sizeof(int));

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(A==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(A);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(f==NULL | g==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(f);

free(g);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MTRand::uint32 oneSeed = 4357UL;

MTRand::uint32 bigSeed[4] = { 0x123, 0x234, 0x345, 0x456 };

MTRand mtrand1( bigSeed, 4 );

MTRand mtrand2( oneSeed );

//###########################################################

for(i=1;i<=nu;i++){

u[i]=0;

u[i+nu]=0;

}

for(i=1;i<=ni;i++){

v[i]=0;

v[i+ni]=0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nr;i++){

fscanf(ff,"%d",&i1); //\*\*\* user

A[i]=i1;

u[i1]++;

fscanf(ff,"%d",&i2); //\*\*\* bus

A[i+nr]=i2;

v[i2]++;

fscanf(ff,"%d\n",&i3); //\*\*\* stars

if(i3<1)i3=1;

else if(i3>5)i3=5;

A[i+2\*nr]=i3;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

n=alfa+1;

for(i=1;i<=n;i++){

for(j=0;j<=1;j++){

g[i][j]=0;

f[i][j]=0;

}

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nr;i++){

i1=A[i]; //\*\*\* user

k=u[i1];

if(k>n)k=n;

i2=A[i+nr]; //\*\*\* bus

k1=v[i2];

if(k1>n)k1=n;

i3=A[i+2\*nr]; //\*\*\* stars

f[k][0]++;

f[k][1]+=i3;

u[i1+nu]+=i3;

g[k1][0]++;

g[k1][1]+=i3;

v[i2+ni]+=i3;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=n;i++){

k=f[i][0];

vs=k;

r1=f[i][1]/vs;

pru[i]=r1;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=3;i<=n;i++){

k=g[i][0];

vs=k;

r1=g[i][1]/vs;

prb[i]=r1;

}

prb[1]=prb[3];

prb[2]=prb[3];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for(i=1;i<=25;i++){

printf("%d %8.6f %d %8.6f %d\n",i,prb[i],g[i][0],pru[i],f[i][0]);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for(i=1;i<=5;i++)hp[i]=0;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nt;i++){

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fscanf(fa,"%d",&i1); //\*\*\* user

fscanf(fa,"%d\n",&i2); //\*\*\* bus

k=u[i1];

if(k<=20){

vs = k;

frac\_user = pow(1.0/(1.0+vs),gamma);

est\_user = pru[k+1];

}

else{

frac\_user = 0.0;

est\_user = 0.0;

}

k1=v[i2];

if(k1<=45){

vs = k1;

frac\_bus = pow(1.0/(1.0+vs),gamma);

est\_bus = prb[k1+1];

}

else{

frac\_bus = 0.0;

est\_bus = 0.0;

}

if(u[i1]>=1 && v[i2]>=1){

q1=u[i1];

q2=v[i2];

p1=(delta+q1)/(q1+q2+2.0\*delta);

p2=(delta+q2)/(q1+q2+2.0\*delta);

r1=(1.0-frac\_user)\*u[i1+nu]/q1 + frac\_user\*est\_user;

r2=(1.0-frac\_bus)\*v[i2+ni]/q2 + frac\_bus\*est\_bus;

q=p1\*log(r1)+p2\*log(r2);

S=exp(q);

q\_cnt=(u[i1]+v[i2])/2.0+2;

hp[1]++;

}

else if(u[i1]>=1){

q1=u[i1];

q2=0.0;

p1=(delta+q1)/(q1+q2+2.0\*delta);

p2=(delta+q2)/(q1+q2+2.0\*delta);

r1=(1.0-frac\_user)\*u[i1+nu]/q1 + frac\_user\*est\_user;

r2=est\_bus;

q=p1\*log(r1)+p2\*log(r2);

S=exp(q);

q\_cnt=u[i1]+2;

hp[2]++;

}

else if(v[i2]>=1){

q1=0.0;

q2=v[i2];

p1=(delta+q1)/(q1+q2+2.0\*delta);

p2=(delta+q2)/(q1+q2+2.0\*delta);

r1=est\_user;

r2=(1.0-frac\_bus)\*v[i2+ni]/q2 + frac\_bus\*est\_bus;

q=p1\*log(r1)+p2\*log(r2);

S=exp(q);

q\_cnt=v[i2]+2;

hp[3]++;

}

else{

q1=0.0;

q2=0.0;

p1=(delta+q1)/(q1+q2+2.0\*delta);

p2=(delta+q2)/(q1+q2+2.0\*delta);

r1=est\_user;

r2=est\_bus;

q=p1\*log(r1)+p2\*log(r2);

S=exp(q);

q\_cnt=2;

hp[4]++;

}

if(S<1.0)S=1.0;

if(S>5.0)S=5.0;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fscanf(fb,"%f",&p1);

if(p1<1.0)p1=1.0;

else if(p1>5.0)p1=5.0;

fscanf(fb,"%f",&q\_count);

if(q\_count<1)q\_count=1.0;

fscanf(fb,"%d\n",&ind1);

vs=(q\_cnt\*log(S)+tau\*q\_count\*log(p1))/(q\_cnt+tau\*q\_count);

q1=exp(vs);

q\_count=q\_cnt+q\_count;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fscanf(fd,"%d",&m);

p1=m;

fscanf(fd,"%d",&m1);

p2=m1;

q\_cnt=(p1+p2)/2.0;

fscanf(fd,"%f\n",&q);

vs=(phi\_gmf\*q\_cnt\*log(q)+q\_count\*log(q1))/(phi\_gmf\*q\_cnt+q\_count);

q1=exp(vs);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fscanf(fc,"%f\n",&p2);

fscanf(fi,"%d",&m);

fscanf(fi,"%d",&m1);

fscanf(fi,"%d\n",&ind);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(ind==0)solut=q1;

else{

r1=pow(10.0/((q\_cnt+q\_count)/2.0+1.0),beta);

if(r1>1)r1=1.0;

solut=r1\*q1+(1.0-r1)\*p2;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fprintf(f1,"%8.6f\n",solut);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fclose(ff);

fclose(fa);

fclose(fb);

fclose(fi);

fclose(fc);

fclose(fd);

fclose(f1);

return(0);

}

########### APPENDIX:: #################################

#!/usr/local/bin/perl

### transform\_d240813.pl ###################

# INPUT: ff) see above (empty space separated); fa) YI nrw; fb) see above (empty space separated);

# fc) see above (TAB separated); f1) YI-user YI-business; ft) see above (empty space separated).

# OUTPUT: fu) seqInd\_user seqInd\_bus Star; fv) seqInd\_user seqInd\_bus Star; fs) statistics regarding availability of indexes.

#===========================================

open(ff, "c:/contest/Kaggle/RecSys/output/train\_user.txt");

open(fa, "c:/contest/Kaggle/RecSys/output\_final/test\_user\_counts\_d230813.txt");

open(fb, "c:/contest/Kaggle/RecSys/output/train\_business.txt");

open(fc, "c:/contest/Kaggle/RecSys/output\_final/test\_bus\_d230813.txt");

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

open(f1, "c:/contest/Kaggle/RecSys/data1/trn\_review.txt");

open(ft, "c:/contest/Kaggle/RecSys/output\_final/tst\_review\_d230813.txt");

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

open(fu, ">c:/contest/Kaggle/RecSys/data\_final/trn\_review\_d240813.txt");

open(fv, ">c:/contest/Kaggle/RecSys/data\_final/tst\_review\_d240813.txt");

open(fs, ">c:/contest/Kaggle/RecSys/data\_final/stats\_d240813.txt");

#===========================================

$\_ = <ff>; #\*\*\* read title

$ii = 0;

while(defined($\_ = <ff>)){

chomp $\_;

$ii++;

@fields = split(/ /,$\_);

$a = @fields[0];

$su{$a} = $ii;

}

#===========================================

$\_ = <fa>; #\*\*\* read title

while(defined($\_ = <fa>)){

chomp $\_;

$ii++;

@fields = split(/ /,$\_);

$a = @fields[0];

$su{$a} = $ii;

}

#===========================================

$\_ = <fb>; #\*\*\* read title

$ii = 0;

while(defined($\_ = <fb>)){

chomp $\_;

$ii++;

@fields = split(/\t/,$\_);

$a = @fields[0];

$sb{$a} = $ii;

}

#===========================================

$\_ = <fc>; #\*\*\* read title

while(defined($\_ = <fc>)){

chomp $\_;

$ii++;

@fields = split(/\t/,$\_);

$a = @fields[0];

$sb{$a} = $ii;

}

#===========================================

$ii=0;

while(defined($\_ = <f1>)){

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

if(defined($su{$a}) && defined($sb{$b})){

$c = @fields[2];

$i1 = $su{$a};

$i2 = $sb{$b};

print fu "$i1 $i2 $c";

}

else{$ii++;}

}

print fs "$ii\n";

#===========================================

$ii=0;

while(defined($\_ = <ft>)){

chomp $\_;

@fields = split(/ /,$\_);

$a = @fields[0];

$b = @fields[1];

if(defined($su{$a}) && defined($sb{$b})){

$i1 = $su{$a};

$i2 = $sb{$b};

print fv "$i1 $i2 1\n";

}

else{

$ii++;

print fv "1000 99 0\n";

}

}

print fs "$ii\n";

#===========================================

close(ff);

close(fa);

close(fb);

close(fc);

close(f1);

close(ft);

close(fu);

close(fv);

close(fs);

########### APPENDIX N2 (calculation of an expected bus-averages):: #####################

#### calculation of an expected user-averages is very similar ##############################

library(gbm)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

A <- read.table("/home/user623/RecSys/rdata\_final/train\_bus\_rdata11.txt", header=TRUE)

T <- read.table("/home/user623/RecSys/rdata\_final/test\_bus\_rdata11.txt", header=TRUE)

##################################################

Target of the code: forecasting of the star-averages (gss\_vsu\_av\_bus\_d240813.txt) for test-business

##################################################

function(n){

alfa = 0.85

m = 400

trace = matrix(0,m+1,4)

nr = nrow(A)

nt = nrow(T)

sol = matrix(0,nt,1)

solut = matrix(0,nr,2)

ind = matrix(0,nr,1)

print("start")

for(jj in 1:m){

for(i in 1:nr){

q = runif(1)

if(q<=alfa){ind[i,1] = 1}

else{ind[i,1] = 0}

}

object <- gbm(tg~.,data=A[ind==1,],distribution="gaussian",n.trees=3000,shrinkage=0.01,interaction.depth=16,n.minobsinnode=5)

sol = sol + predict(object,newdata=T,n.trees=3000,type="response")

tr = predict(object,newdata=A,n.trees=3000,type="response")

S = 0.0

k = 0

S1 = 0.0

k1 = 0

for(i in 1:nr){

if(tr[i]<1.0){tr[i] = 1.0}

else if(tr[i]>5.0){tr[i] = 5.0}

if(ind[i,1] == 1){

S1 = S1 + (A[i,1] - tr[i])^2.0

k1 = k1 + 1

}

else{

solut[i,1] = solut[i,1] + 1

solut[i,2] = solut[i,2] + tr[i]

S = S + (A[i,1] - tr[i])^2.0

k = k + 1

}

}

trace[jj,1] = jj

trace[jj,2] = k/nr

trace[jj,3] = sqrt(S/k)

trace[jj,4] = sqrt(S1/k1)

print(trace[jj,])

}

S = 0.0

i1 = 0

for(i in 1:nr){

if(solut[i,1]>=1){

i1 = i1+1

if(solut[i,1] >= 2){

solut[i,2] = solut[i,2]/solut[i,1]

}

if(solut[i,2]<1.0){solut[i,2] = 1.0}

else if(solut[i,2]>5.0){solut[i,2] = 5.0}

S = S + (A[i,1] - solut[i,2])^2.0

}

}

trace[m+1,1] = jj

trace[m+1,2] = i1

trace[m+1,3] = sqrt(S/i1)

print(trace[m+1,])

sol <- sol/m

write.table(trace, file = "/home/user623/RecSys/output\_final/trace\_vsu\_av\_bus\_d240813.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

write.table(sol, file = "/home/user623/RecSys/output\_final/gss\_vsu\_av\_bus\_d240813.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

}

########################################################

library(gbm)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

A <- read.table("c:/contest/Kaggle/RecSys/rdata\_final/train\_user\_rdata11.txt", header=TRUE)

T <- read.table("c:/contest/Kaggle/RecSys/rdata\_final/test\_user\_rdata11.txt", header=TRUE)

##################################################

Analogous to the above code: forecasting of the star-averages (gss\_vsu\_av\_bus\_d240813.txt) for test-users

##################################################

function(n){

alfa = 0.55

m = 50

trace = matrix(0,m+1,4)

nr = nrow(A)

nt = nrow(T)

sol = matrix(0,nt,1)

solut = matrix(0,nr,2)

ind = matrix(0,nr,1)

print("start")

for(jj in 1:m){

for(i in 1:nr){

q = runif(1)

if(q<=alfa){ind[i,1] = 1}

else{ind[i,1] = 0}

}

object <- gbm(tg~.,data=A[ind==1,],distribution="gaussian",n.trees=800,shrinkage=0.01,interaction.depth=16,n.minobsinnode=5)

sol = sol + predict(object,newdata=T,n.trees=800,type="response")

tr = predict(object,newdata=A,n.trees=800,type="response")

S = 0.0

k = 0

S1 = 0.0

k1 = 0

for(i in 1:nr){

if(tr[i]<1.0){tr[i] = 1.0}

else if(tr[i]>5.0){tr[i] = 5.0}

if(ind[i,1] == 1){

S1 = S1 + (A[i,1] - tr[i])^2.0

k1 = k1 + 1

}

else{

solut[i,1] = solut[i,1] + 1

solut[i,2] = solut[i,2] + tr[i]

S = S + (A[i,1] - tr[i])^2.0

k = k + 1

}

}

trace[jj,1] = jj

trace[jj,2] = k/nr

trace[jj,3] = sqrt(S/k)

trace[jj,4] = sqrt(S1/k1)

print(trace[jj,])

}

S = 0.0

i1 = 0

for(i in 1:nr){

if(solut[i,1]>=1){

i1 = i1+1

if(solut[i,1] >= 2){

solut[i,2] = solut[i,2]/solut[i,1]

}

if(solut[i,2]<1.0){solut[i,2] = 1.0}

else if(solut[i,2]>5.0){solut[i,2] = 5.0}

S = S + (A[i,1] - solut[i,2])^2.0

}

}

trace[m+1,1] = jj

trace[m+1,2] = i1

trace[m+1,3] = sqrt(S/i1)

print(trace[m+1,])

sol <- sol/m

write.table(trace, file = "c:/contest/Kaggle/RecSys/output\_final/trace\_av\_user\_d240813.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

write.table(sol, file = "c:/contest/Kaggle/RecSys/output\_final/gss\_av\_user\_d240813.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

}

//\*\*\*\* transf\_bus\_rdata\_d240813.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\* RecSys 2013 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\* preparation of the data for above R-code (case of business) with 1) counts; 2) latitude; 3) longitude; 4) 10 indexes of the most important //\*\*\* categories; 5) number of the most important categories, 6) nrw.

//\*\*\* Remark: this is just an illustration. We can add in the field of features blocks corresponding to votes and checkins.

//###########################################################

#include<math.h>

#include<stdlib.h>

#include<stdio.h>

main()

{

int i,ii,j;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int nr=11537; //\*\*\* training

const int nt=2797;

const int nc=11;

//================================================

float vs,q,p;

//================================================

FILE \*ff,\*fa,\*fb,\*fc,\*ft,\*f1,\*f2;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ff=fopen("c:/contest/Kaggle/RecSys/rdata\_final/train\_bus\_tr11.txt","r");

fa=fopen("c:/contest/Kaggle/RecSys/rdata\_final/test\_bus\_tr11.txt","r");

fb=fopen("c:/contest/Kaggle/RecSys/output1/long\_lat\_business.txt","r");

fc=fopen("c:/contest/Kaggle/RecSys/output1/stars\_count\_business.txt","r");

ft=fopen("c:/contest/Kaggle/RecSys/rdata\_final/long\_lat\_count\_bus\_tst.txt","r");

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// mkdir("c:/contest/Kaggle/RecSys/rdata\_final/");

f1=fopen("c:/contest/Kaggle/RecSys/rdata\_final/train\_bus\_rdata11.txt","w");

f2=fopen("c:/contest/Kaggle/RecSys/rdata\_final/test\_bus\_rdata11.txt","w");

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fprintf(f1,"tg ");

for(i=1;i<=nc+2;i++){

fprintf(f1,"v%d ",i);

fprintf(f2,"v%d ",i);

}

fprintf(f1,"v%d\n",nc+3);

fprintf(f2,"v%d\n",nc+3);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nr;i++){

fscanf(fc,"%f",&vs);

fscanf(fc,"%d\n",&ii);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fprintf(f1,"%2.1f %d ",vs,ii);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fscanf(fb,"%f",&p);

fscanf(fb,"%f\n",&q);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fprintf(f1,"%6.4f %6.4f ",p,q);

for(j=1;j<=nc-1;j++){

fscanf(ff,"%d",&ii);

fprintf(f1,"%d ",ii);

}

fscanf(ff,"%d\n",&ii);

fprintf(f1,"%d\n",ii);

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nt;i++){

fscanf(ft,"%f",&p);

fscanf(ft,"%f",&q);

fscanf(ft,"%d\n",&ii);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fprintf(f2,"%d %6.4f %6.4f ",ii,p,q);

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(j=1;j<=nc-1;j++){

fscanf(fa,"%d",&ii);

fprintf(f2,"%d ",ii);

}

fscanf(fa,"%d\n",&ii);

fprintf(f2,"%d\n",ii);

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fclose(ff);

fclose(fa);

fclose(fb);

fclose(fc);

fclose(ft);

fclose(f1);

fclose(f2);

return(0);

}

########################################################

%\*\*\* ../RecSys/transf\_d240813a.m \*\*\*\*\*\*\*\*\*\*\*

%\*\*\* Dimensional reduction (novel format for categorical variables)

%\*\*\* Transformation from binary format of 354 categories to only 11 (see Section 2.2 of the paper)

%=========================================

clear;

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

f1=fopen('c:/contest/Kaggle/RecSys/rdata\_final/train\_bus\_tr11.txt','w');

f2=fopen('c:/contest/Kaggle/RecSys/rdata\_final/test\_bus\_tr11.txt','w');

f3=fopen('c:/contest/Kaggle/RecSys/rdata\_final/train\_user\_tr11.txt','w');

f4=fopen('c:/contest/Kaggle/RecSys/rdata\_final/test\_user\_tr11.txt','w');

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

A=load('c:/contest/Kaggle/RecSys/rdata\_final/train\_bus\_rdata354\_d240813.txt');

B=load('c:/contest/Kaggle/RecSys/rdata\_final/test\_bus\_rdata354\_d240813.txt');

A1=load('c:/contest/Kaggle/RecSys/rdata\_final/train\_user\_rdata354\_d240813.txt');

B1=load('c:/contest/Kaggle/RecSys/rdata\_final/test\_user\_rdata354\_d240813.txt');

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

r=sum(A);

m=size(A,2);

k=10;

n=size(A,1);

A\_trn=zeros(n,k+1);

for i=1:size(A,1),

v=A(i,:);

for j=1:size(A,2),

if(A(i,j)>0)

v(j)=r(j);

end;

end;

[dir,I]=sort(v);

i2=0;

i3=0;

for j=1:k,

ii=I(m-j+1);

if(A(i,ii)>0)

A\_trn(i,j)=ii;

i2=ii;

i3=i3+1;

else

A\_trn(i,j)=i2;

end;

fprintf(f1,'%d ',A\_trn(i,j));

end;

A\_trn(i,k+1)=i3;

fprintf(f1,'%d\n',A\_trn(i,k+1));

end;

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

n=size(B,1);

A\_trn=zeros(n,k+1);

for i=1:size(B,1),

v=B(i,:);

for j=1:size(B,2),

if(B(i,j)>0)

v(j)=r(j);

end;

end;

[dir,I]=sort(v);

i2=0;

i3=0;

for j=1:k,

ii=I(m-j+1);

if(B(i,ii)>0)

A\_trn(i,j)=ii;

i2=ii;

i3=i3+1;

else

A\_trn(i,j)=i2;

end;

fprintf(f2,'%d ',A\_trn(i,j));

end;

A\_trn(i,k+1)=i3;

fprintf(f2,'%d\n',A\_trn(i,k+1));

end;

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

n=size(A1,1);

A\_trn=zeros(n,k+1);

for i=1:size(A1,1),

v=A1(i,:);

for j=1:size(A1,2),

if(A1(i,j)>0)

v(j)=r(j);

end;

end;

[dir,I]=sort(v);

i2=0;

i3=0;

for j=1:k,

ii=I(m-j+1);

if(A1(i,ii)>0)

A\_trn(i,j)=ii;

i2=ii;

i3=i3+1;

else

A\_trn(i,j)=i2;

end;

fprintf(f3,'%d ',A\_trn(i,j));

end;

A\_trn(i,k+1)=i3;

fprintf(f3,'%d\n',A\_trn(i,k+1));

end;

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

n=size(B1,1);

A\_trn=zeros(n,k+1);

for i=1:size(B1,1),

v=B1(i,:);

for j=1:size(B1,2),

if(B1(i,j)>0)

v(j)=r(j);

end;

end;

[dir,I]=sort(v);

i2=0;

i3=0;

for j=1:k,

ii=I(m-j+1);

if(B1(i,ii)>0)

A\_trn(i,j)=ii;

i2=ii;

i3=i3+1;

else

A\_trn(i,j)=i2;

end;

fprintf(f4,'%d ',A\_trn(i,j));

end;

A\_trn(i,k+1)=i3;

fprintf(f4,'%d\n',A\_trn(i,k+1));

end;

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fclose('all');

########################################################

//\*\*\*\* transf\_bus\_d240813.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\* RecSys 2013 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\* preparation of input binary data for the above MATLAB code (business)

//###########################################################

#include<math.h>

#include<stdlib.h>

#include<stdio.h>

main()

{

int i,i1,j,k,h[19];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int nr=11537; //\*\*\* training

const int nt=2797;

const int nc=354;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

int \*u;

//================================================

FILE \*ff,\*fa,\*f1,\*f2;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ff=fopen("c:/contest/Kaggle/RecSys/output5/train\_bus\_transf.txt","r");

fa=fopen("c:/contest/Kaggle/RecSys/output\_final/test\_bus\_d240813.txt","r");

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

f1=fopen("c:/contest/Kaggle/RecSys/rdata\_final/train\_bus\_rdata354\_d240813.txt","w");

f2=fopen("c:/contest/Kaggle/RecSys/rdata\_final/test\_bus\_rdata354\_d240813.txt","w");

//++++++++++++++++++++++++++++++++++++++++++++++++

u = (int \*)calloc(501, sizeof(int));

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(u==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(u);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nc;i++)u[i]=0;

for(i=1;i<=nr;i++){

fscanf(ff,"%d",&k);

if(k>=1){

for(j=1;j<=k;j++){

fscanf(ff,"%d",&i1);

h[j]=i1;

u[i1]=1;

}

}

fscanf(ff,"\n");

for(j=1;j<=nc-1;j++)fprintf(f1,"%d ",u[j]);

fprintf(f1,"%d\n",u[nc]);

if(k>=1){

for(j=1;j<=k;j++){

i1=h[j];

u[i1]=0;

}

}

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nt;i++){

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fscanf(fa,"%d",&k);

if(k>=1){

for(j=1;j<=k;j++){

fscanf(fa,"%d",&i1);

h[j]=i1;

u[i1]=1;

}

}

fscanf(fa,"\n");

for(j=1;j<=nc-1;j++)fprintf(f2,"%d ",u[j]);

fprintf(f2,"%d\n",u[nc]);

if(k>=1){

for(j=1;j<=k;j++){

i1=h[j];

u[i1]=0;

}

}

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fclose(ff);

fclose(fa);

fclose(f1);

fclose(f2);

return(0);

}

########################################################

//\*\*\*\* transf\_user\_d240813.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\* RecSys 2013 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\* preparation of input binary data (case of user)

//#######################################################

#include<math.h>

#include<stdlib.h>

#include<stdio.h>

main()

{

int i,i1,ii,j,j2,k,k1,m;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int nr=11537; //\*\*\* training

const int nt=2797;

const int nc=354;

//================================================

const int n1=43873;

const int n2=9522;

//================================================

float p;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

int \*A,\*v;

//================================================

FILE \*ff,\*fa,\*gg,\*ga,\*f1,\*f2;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ff=fopen("c:/contest/Kaggle/RecSys/output5/train\_bus\_transf.txt","r");

fa=fopen("c:/contest/Kaggle/RecSys/output\_final/test\_bus\_d240813.txt","r");

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

gg=fopen("c:/contest/Kaggle/RecSys/output1/train\_user\_transf.txt","r");

ga=fopen("c:/contest/Kaggle/RecSys/output\_final/test\_user\_transf\_d240813.txt","r");

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

f1=fopen("c:/contest/Kaggle/RecSys/rdata\_final/train\_user\_rdata354\_d240813.txt","w");

f2=fopen("c:/contest/Kaggle/RecSys/rdata\_final/test\_user\_rdata354\_d240813.txt","w");

//++++++++++++++++++++++++++++++++++++++++++++++++

A = (int \*)calloc(1000001, sizeof(int));

v = (int \*)calloc(2001, sizeof(int));

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(A==NULL | v==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(A);

free(v);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

m=nr+nt;

for(i=1;i<=nr;i++){

fscanf(ff,"%d",&k);

A[i]=k;

if(k==0)fscanf(ff,"\n");

else{

for(j=1;j<=k;j++){

fscanf(ff,"%d",&i1);

A[i+m\*j]=i1;

}

fscanf(ff,"\n");

}

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nt;i++){

fscanf(fa,"%d",&k);

A[nr+i]=k;

if(k==0)fscanf(fa,"\n");

else{

for(j=1;j<=k;j++){

fscanf(fa,"%d",&i1);

A[nr+i+m\*j]=i1;

}

fscanf(fa,"\n");

}

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=n1;i++){

for(j=1;j<=nc;j++)v[j]=0;

fscanf(gg,"%d",&k);

p=k;

for(j=1;j<=k;j++){

fscanf(gg,"%d",&i1);

k1=A[i1];

if(k1>=1){

for(j2=1;j2<=k1;j2++){

ii=A[i1+m\*j2];

v[ii]++;

}

}

}

fscanf(gg,"\n");

for(j=1;j<=nc;j++){

if(v[j]>=1)fprintf(f1,"%6.4f ",v[j]/p);

else fprintf(f1,"%d ",0);

}

fprintf(f1,"\n");

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=n2;i++){

for(j=1;j<=nc;j++)v[j]=0;

fscanf(ga,"%d",&k);

p=k;

for(j=1;j<=k;j++){

fscanf(ga,"%d",&i1);

k1=A[i1];

if(k1>=1){

for(j2=1;j2<=k1;j2++){

ii=A[i1+m\*j2];

v[ii]++;

}

}

}

fscanf(ga,"\n");

for(j=1;j<=nc;j++){

if(v[j]>=1)fprintf(f2,"%6.4f ",v[j]/p);

else fprintf(f2,"%d ",0);

}

fprintf(f2,"\n");

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fclose(ff);

fclose(fa);

fclose(gg);

fclose(ga);

fclose(f1);

fclose(f2);

return(0);

}

########################################################

#!/usr/local/bin/perl

### transf\_business\_d240813.pl #############

#===========================================

open(ff, "c:/contest/Kaggle/RecSys/output/train\_business2.txt");

open(f1, "c:/contest/Kaggle/RecSys/output\_final/test\_business2.txt");

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

open(fa, ">c:/contest/Kaggle/RecSys/output\_final/train\_bus\_d240813.txt");

open(fb, ">c:/contest/Kaggle/RecSys/output\_final/test\_bus\_d240813.txt");

open(fc, ">c:/contest/Kaggle/RecSys/output\_final/stats\_bus\_d240813.txt");

#===========================================

$delta = 5;

$ii = 0;

while(defined($\_ = <ff>)){

chomp $\_;

@fields = split(/\t/,$\_);

$n = @fields;

if($n>=1){

$n1 = $n - 1;

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sf{$a})){$sf{$a}++;}

else{

$ii++;

$sa{$a} = $ii;

$sf{$a} = 1;

}

}

}

}

#===========================================

$ii = 0;

foreach $elem (keys %sa){

if($sf{$elem}>=$delta){

$ii++;

$sb{$elem} = $ii;

}

}

print fc "$ii\n";

#===========================================

close(ff);

open(ff, "c:/contest/Kaggle/RecSys/output/train\_business2.txt");

#===========================================

while(defined($\_ = <ff>)){

chomp $\_;

@fields = split(/\t/,$\_);

$n = @fields;

$m = 0;

if($n>=1){

$n1 = $n - 1;

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$m++;

}

}

}

if($m==0){print fa "$m\n";}

else{

print fa "$m ";

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$x = $sb{$a};

print fa "$x ";

}

}

print fa "\n";

}

}

#===========================================

while(defined($\_ = <f1>)){

chomp $\_;

@fields = split(/\t/,$\_);

$n = @fields;

$m = 0;

if($n>=1){

$n1 = $n - 1;

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$m++;

}

}

}

if($m==0){print fb "$m\n";}

else{

print fb "$m ";

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$x = $sb{$a};

print fb "$x ";

}

}

print fb "\n";

}

}

#===========================================

close(ff);

close(f1);

close(fa);

close(fb);

close(fc);

########################################################

#!/usr/local/bin/perl

### transfer\_bus\_tst\_d230813.pl ############

# Remark: calculation of the train\_business2.txt is about the same

#===========================================

open(ff, "c:/contest/Kaggle/RecSys/output\_final/test\_business1.txt");

#mkdir("c:/contest/Kaggle/RecSys/output/");

open(fa, ">c:/contest/Kaggle/RecSys/output\_final/test\_business2.txt");

open(fb, ">c:/contest/Kaggle/RecSys/output\_final/num\_business\_tst.txt");

#===========================================

$a = 2;

$n = 0;

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

$\_ = <ff>;

while(defined($\_ = <ff>)){

chomp $\_;

if($\_ ne $a){

$n++;

print fa "$\_\t";

}

else{

print fa "\n";

print fb "$n\n";

$n = 0;

$a++;

}

}

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

close(ff);

close(fa);

close(fb);

########################################################

library(rjson)

# Convert raw JSON file into managable data frame

UnpackJSON <- function(filePath){

con <- file(filePath, "r")

input <- readLines(con)

jsonData <- sapply(input,fromJSON)

close(con)

df <- data.frame(jsonData)

temp <- rownames(df)

df <- as.data.frame(t(df))

colnames(df) <- temp

rownames(df) <- NULL

return(df)

}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Convert the nested lists into regular vectors

UnlistJSON <- function(A){

for(i in 1:nrow(A)){

q = unlist(A[i,2])

write(i, file = "c:/contest/Kaggle/RecSys/output\_final/test\_business1.txt", append=TRUE, sep = "#")

write(q, file = "c:/contest/Kaggle/RecSys/output\_final/test\_business1.txt", append=TRUE, sep = "\t")

}

}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Training User data

filePath <- "c:/contest/Kaggle/RecSys/final\_data/final\_test\_set\_business.json"

user <- UnpackJSON(filePath)

user <- user[,c('type', 'categories')]

UnlistJSON(user)

########################################################

#!/usr/local/bin/perl

### transf\_business1.pl ####################

#===========================================

open(ff, "c:/contest/Kaggle/RecSys/output/train\_business2.txt");

open(f1, "c:/contest/Kaggle/RecSys/output/test\_business2.txt");

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mkdir("c:/contest/Kaggle/RecSys/output5/");

open(fa, ">c:/contest/Kaggle/RecSys/output5/train\_bus\_transf.txt");

open(fb, ">c:/contest/Kaggle/RecSys/output5/test\_bus\_transf.txt");

open(fc, ">c:/contest/Kaggle/RecSys/output5/stats\_bus\_transf.txt");

#===========================================

$delta = 5;

$ii = 0;

while(defined($\_ = <ff>)){

chomp $\_;

@fields = split(/\t/,$\_);

$n = @fields;

if($n>=1){

$n1 = $n - 1;

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sf{$a})){$sf{$a}++;}

else{

$ii++;

$sa{$a} = $ii;

$sf{$a} = 1;

}

}

}

}

#===========================================

$ii = 0;

foreach $elem (keys %sa){

if($sf{$elem}>=$delta){

$ii++;

$sb{$elem} = $ii;

}

}

print fc "$ii\n";

#===========================================

close(ff);

open(ff, "c:/contest/Kaggle/RecSys/output/train\_business2.txt");

#===========================================

while(defined($\_ = <ff>)){

chomp $\_;

@fields = split(/\t/,$\_);

$n = @fields;

$m = 0;

if($n>=1){

$n1 = $n - 1;

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$m++;

}

}

}

if($m==0){print fa "$m\n";}

else{

print fa "$m ";

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$x = $sb{$a};

print fa "$x ";

}

}

print fa "\n";

}

}

#===========================================

while(defined($\_ = <f1>)){

chomp $\_;

@fields = split(/\t/,$\_);

$n = @fields;

$m = 0;

if($n>=1){

$n1 = $n - 1;

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$m++;

}

}

}

if($m==0){print fb "$m\n";}

else{

print fb "$m ";

for($i=0;$i<=$n1;$i++){

$a = @fields[$i];

if(defined($sb{$a})){

$x = $sb{$a};

print fb "$x ";

}

}

print fb "\n";

}

}

#===========================================

close(ff);

close(f1);

close(fa);

close(fb);

close(fc);

########################################################