HMRF_EM Report for J_Martin

group 1

April 15, 2017

Step 0: Load the packages, specify directories

```
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(text2vec, dplyr, qlcMatrix, kernlab, knitr)
library("stringr")
library("gtools")
setwd("~/Documents/GitHub/Spr2017-proj4-team1/data")
# here replace it with your own path or manually set it in RStudio
# to where this rmd file is located
```

Step 1: Load and process the data

```
#get author_id, paper_id, coauthor_list, paper_title, journal name
data.lib="~/Documents/GitHub/Spr2017-proj4-team1/data/nameset"
data.files=list.files(path=data.lib, "*.txt")
data.files
```

```
##  [1] "AGupta.txt"    "AKumar.txt"    "CChen.txt"     "DJohnson.txt"
##  [5] "JLee.txt"      "JMartin.txt"   "JRobinson.txt" "JSmith.txt"
##  [9] "KTanaka.txt"   "MBrown.txt"    "MJones.txt"    "MMiller.txt"
## [13] "SLee.txt"      "YChen.txt"
```

```
### remove "*.txt"
query.list=substring(data.files,
                     1, nchar(data.files)-4)

query.list
```

```
##  [1] "AGupta"    "AKumar"    "CChen"     "DJohnson"  "JLee"
##  [6] "JMartin"   "JRobinson" "JSmith"    "KTanaka"   "MBrown"
## [11] "MJones"    "MMiller"   "SLee"      "YChen"
```

```
## add a space
query.list=paste(substring(query.list, 1, 1),
                 " ",
                 substring(query.list,
                           2, nchar(query.list)),
                 sep=""
)

query.list
```

```
##  [1] "A Gupta"    "A Kumar"    "C Chen"     "D Johnson"  "J Lee"
##  [6] "J Martin"   "J Robinson" "J Smith"    "K Tanaka"   "M Brown"
## [11] "M Jones"    "M Miller"   "S Lee"      "Y Chen"
```

```
# Write a function to get the list of author_id, paper_id, coauthor_list, paper_title, journal name

f.line.proc=function(lin, nam.query="."){

  # remove unwanted characters
  char_notallowed <- "\\@#$%^&?"
  lin.str=str_replace(lin, char_notallowed, "")

  # get author id
  lin.str=strsplit(lin.str, "_")[[1]]
  author_id=as.numeric(lin.str[1])

  # get paper id
  lin.str=lin.str[2]
  paper_id=strsplit(lin.str, " ")[[1]][1]
  lin.str=substring(lin.str, nchar(paper_id)+1, nchar(lin.str))
  paper_id=as.numeric(paper_id)

  # get coauthor list
  lin.str=strsplit(lin.str, "<>")[[1]]
  coauthor_list=strsplit(lin.str[1], ";")[[1]]

  #print(lin.str)
  for(j in 1:length(coauthor_list)){
    if(nchar(coauthor_list[j])>0){
      nam = strsplit(coauthor_list[j], " ")[[1]]
      if(nchar(nam[1])>0){
        first.ini=substring(nam[1], 1, 1)
      }else{
        first.ini=substring(nam[2], 1, 1)
      }
    }
    last.name=nam[length(nam)]
    nam.str = paste(first.ini, last.name)
    coauthor_list[j]=nam.str
  }

  match_ind = charmatch(nam.query, coauthor_list, nomatch=-1)

  #print(nam.query)
  #print(coauthor_list)
  #print(match_ind)

  if(match_ind>0){

    coauthor_list=coauthor_list[-match_ind]
  }

  paper_title=lin.str[2]
  journal_name=lin.str[3]
```

```
  list(author_id=author_id,
       paper_id=paper_id,
       coauthor_list=coauthor_list,
       paper_title=paper_title,
       journal_name=journal_name)
}

#read data
data_list=list(1:length(data.files))

for(i in 1:length(data.files)){

  dat=as.list(readLines(paste(data.lib, data.files[i], sep="/")))
  data_list[[i]]=lapply(dat, f.line.proc, nam.query=query.list[i])


}
names(data_list)=query.list
```

Step 2: Feature design

Let's first create a vocabulary-based DTM. Here we collect unique terms from all documents and mark each of them with a unique ID using the `create_vocabulary()` function. We use an iterator to create the vocabulary.

```
it_train_list <- list(1:length(data.files))
vocab <- list(1:length(data.files))
author_id<-list(1:length(data.files))
for (j in 1:length(data.files)) {
  data_unlist <- unlist(data_list[[j]])
  paper_title<- as.vector(data_unlist[which(names(data_unlist)=="paper_title")])
  paper_id<- as.vector(data_unlist[which(names(data_unlist)=="paper_id")])
  author_id[[j]]<- as.vector(data_unlist[which(names(data_unlist)=="author_id")])
  it_train_list[[j]] <- itoken(paper_title,
             preprocessor = tolower,
             tokenizer = word_tokenizer,
             ids =paper_id,
             progressbar = FALSE)
vocab[[j]] <- create_vocabulary(it_train_list[[j]], stopwords = c("a", "an", "the", "in", "on",
                                                  "at", "of", "above", "under"))

vocab[[j]]


}
```

Here, we remove pre-defined stopwords, the words like ??a??, ??the??, ??in??, ??I??, ??you??, ??on??, etc, which do not provide much useful information.

Now that we have a vocabulary list, we can construct a document-term matrix.

```
#construct DTM
vectorizer <- list(1:length(data.files))
dtm_train <- list(1:length(data.files))
for(i in 1:length(data.files)){
  vectorizer[[i]] <- vocab_vectorizer(vocab[[i]])
  dtm_train[[i]] <- create_dtm(it_train_list[[i]], vectorizer[[i]])
}
```

Now we have DTM and can check its dimensions.

```
for (i in 1:length(data.files)){
print(dim(dtm_train[[i]]))
}
```

```
## [1]  577 1261
## [1] 244 666
## [1]  801 1762
## [1] 368 874
## [1] 1419 2448
## [1] 112 435
## [1] 171 541
## [1]  927 1787
## [1] 280 828
## [1] 153 485
## [1] 260 692
## [1]  412 1001
## [1] 1464 2238
## [1] 1265 2188
```

Then, we want to use DTM to compute TF-IDF transformation on DTM.

```
dtm_train_tfidf  <- list(1:length(data.files))

for(i in 1:length(data.files)){
tfidf <- TfIdf$new()
dtm_train_tfidf[[i]] <- fit_transform(dtm_train[[i]], tfidf)
}
```

Step 3: Write HMRF EM Algorithm

We will only use dataset 6 for testing! Define distance function D(xi,xj)

```
# This function compute the distance as described in the paper using a (squared) matrix A and two vecto
distance = function(A,xi,xj){
  A = as.matrix(A)
  xi = as.matrix(xi)
  xj = as.matrix(xj)
  normxi = sqrt(abs(crossprod(xi,A)%*% xi))[1]
  normxj = sqrt(abs(crossprod(xj,A)%*% xj))[1]
  return(1 - (crossprod(xi,A) %*% xj)[1]/(normxi*normxj))
}
#test: obj = distance(A,X_paper[24,],Y[1,])
```

Define contraints 2& 6

```
M <- list(1:length(data.files))

d=6
  #calculate n and p
  n <- length(data_list[[d]])
  group <- data_list[[d]]
  coauthor_list <- vector(length=0)
  for(j in 1:n){
    coauthor_list_add <- group[[j]]$coauthor_list
    coauthor_list <-c(coauthor_list,coauthor_list_add)
  }
  coauthor_list <- unique(coauthor_list)
  p <- length(coauthor_list)

  #construct submatrix
  #construct Mp
  Mp <- diag(n)
  #construct Map
  Map <- Matrix(rep(0,n*p),p,n)
  for(j in 1:p){
    for(i in 1:n){
       Map[j,i] =ifelse(coauthor_list[j] %in% group[[i]]$coauthor_list,1,0)
       #whether aj is the co-author in pi
    }
        }
  #transform the Matrix
  #construct Mpa
  Mpa <- t(Map)

  #construct Ma(let's limit our database in the data_list)
  Ma <- Matrix(rep(0,p^2),p,p)
  for (i in 1:p){
    for(j in 1:p){
      #Ma[i,j]=1 indicates coauthor_i and coauthor_j in the same publication
      #if both Map[i,m]=1 & Mpa[m,j]=1, Ma[i,j]=1
         Ma[i,j]=ifelse(Map[i,]%*%Mpa[,j]>0,1,0)
    }
  }


  M[[d]] <- cbind(rbind(Mp,Map),rbind(Mpa,Ma))


M1 <- list(1:length(data.files))

M1[[d]] <- M[[d]]%*%M[[d]]


##Assume t=2, then we get 2-coauthor constraint
M2 <- list(1:length(data.files))
M2[[d]] <- M[[d]]%*%M[[d]]%*%M[[d]]
```

```
constraint2 <-function(i,j,d){
  c6<-ifelse(M1[[d]][i,j]>=1,1,0)
  return(c6)
}

constraint6 <-function(i,j,d){
  c6<-ifelse(M2[[d]][i,j]>=1,1,0)
  return(c6)
}

#We only tried until 3, which is enough. We get a tao, which is a matrix between papers
```

Objective Function

```
# This function compute the objective function at the point defined by the distance matrix D , the vect

#number of clusters:
#k=length(unique(author_id))

objective = function(Y, i,h, d, w2 = 0.7, w6 = 0.7^2){

  X_paper=as.matrix(dtm_train_tfidf[[d]])
  X_paper1=X_paper[l!=h]
  obj = distance(A,X_paper[i,],Y[h,])

    for(j in nrow(X_paper1)){
      obj = obj + distance(A,X_paper1[i,],X_paper1[j,]) * (w2 * constraint2(i,j,d) + w6 * constraint6(i
    }

  return(obj)
}
```

Hmrf-em algorithm #initiialization

```
#Initialization
Initialization <-function(A,d,k){
  #create index set
  index=1:dim(dtm_train_tfidf[[d]])[1]
  #define label
  l=vector()
  #initial label
  lambda=0
  index_new=index

  while(length(index)>0){
     lambda=lambda+1
     #print(paste("lambda=",lambda))

    for (j in index){
    i=index[1]
    if(constraint2(i,j,d)>0|constraint6(i,j,d)>0)
      {
    dele = which(index_new==j)
```

```
  index_new <- index_new[-dele]#remove index already belong to a cluster
  l[j]=lambda
   }
  }
  index=index_new
  #print(paste("index=",index[1],"length=",length(index)))
}

if(lambda<k){

    for(i in 1:lambda){
      #calculate centroids
  X = as.matrix(dtm_train_tfidf[[d]])[l==i,] #subset the Xi to the one with the label
  Y=matrix(NA,lambda,dim(dtm_train_tfidf[[d]])[2])
  Y[i,] = Y[i,]/(sqrt((t(Y[i,]) %*% A %*% Y[i,])[1]))
    }
  glo_Y = rowSums(Y)
  glo_Y =glo_Y/(sqrt((t(glo_Y) %*% A %*% glo_Y)[1]))
  while(k-lambda>0){
    lambda=lambda+1
    Y[lambda,]=glo_Y+rnorm(n=dim(dtm_train_tfidf[[d]])[2],mean=0,sd=(3)^(1/3))
  }
  l=Estep(A, l, Y, d, k , w2 = 0.7, w6 = 0.7^2)
  }


else if(lambda>k){

  X = as.matrix(dtm_train_tfidf[[d]])
  #calculate the distances between clusters and combine the closest together
 #too slow so deciede to use kmeans
  #while(lambda-k>0){
     Y=matrix(NA,lambda,dim(dtm_train_tfidf[[d]])[2])
    for(i in 1:lambda){
      #calculate centroids
  Z=as.matrix(X[l==i,],ncol=dim(dtm_train_tfidf[[d]])[2])#subset the Xi to the one with the label
  Y[i,] = colSums(Z)
  Y[i,] = Y[i,]/(crossprod(Y[i,],A)%*% Y[i,])[1]
    }
    cluster_label=kmeans(Y,centers=k)$cluster
    #re-order the cluster number


    uni=unique(cluster_label)
    for(j in 1:k){
      loc=which(cluster_label==uni[j])

      for(i in loc){
      l[l==i]=j
      }
    }
```

```
      #calculate distances between cluster j and cluster h
       #we create an upper matrix. we assign half of the values as a very large number.
     # dis=matrix(100,lambda,lambda)
      #for(j in 1:lambda){
      #  for(h in (j+1):lambda){
      #    dis[j,h]=distance(A,Y[j,],Y[h,])
      #    print(dis[j,h])
      #  }
    #  }
      #find the closest two clusters and assign the larger index cluster to the lower index
    #k1=which(dis==min(dis),arr.ind=TRUE)[1]
    #k2=which(dis==min(dis),arr.ind=TRUE)[2]
    #l[l==max(k1,k2)]=min(k1,k2)
    #lambda=lambda-1
   # }
  }
  return(l)
}
```

```
# - first clustering respecting the constraints c2 and c6 in k groups ; and A = identity
# can do using constraint2 and constraint6
#we need to define A
#E-step
Estep = function(A, l, Y, d, w2 = 0.7, w6 = w2^2){
  #first generate the random orderand loop over them
 random_order = sample(1:length(l),length(l))
  for(i in 1:length(l)){
    # now we are going to compute all f ( y h , x i )  and take the min of it

    f2=vector()
    for(j in 1:k){
      f2[j] = objective(Y=Y, i=i,  h=j, d=d)
      }
    l[i] = which.min(f2)
  }
  return(l)
}



#Mstep
Mstep = function(A, l, Y, d,w2 = 0.7, w6 = w2^2, eta=0.01){
  #update Y first:
  Y=matrix(NA,k,dim(dtm_train_tfidf[[d]])[2])
  for(j in 1:k){
    X = as.matrix(dtm_train_tfidf[[d]])
    Z=as.matrix(X[l==j,],ncol=dim(dtm_train_tfidf[[d]])[2])#subset the Xi to the one with the label
    Y[j,] = col_sums(Z)
    Y[j,] = Y[j,]/(crossprod(Y[j,],A)%*% Y[j,])[1]
  }


  #update the distance matrix now
```

```
  m = dim(A)[1]
  Anew = diag(rep(1,m))
  Add=rep(0,m)
  data=as.matrix(dtm_train_tfidf[[d]])
    #compute norm matrix
  norm=abs(tcrossprod((data%*%A),data))
  sqr_norm=sqrt(norm)

    for(i in 1:(length(l)-1)){

      xi = data[i,]
      y=Y[l[i],]
      normxi = sqr_norm[i,i]
      normxiy=abs((crossprod(xi,A)%*% y)[1])
      normy=sqrt(abs(crossprod(y,A)%*% y))[1]
    partial_D_xiy = (xi*y)/(normxi*normy)-(xi*xi)*normxiy/(2*normxi*normy^3)-(y*y)*normxiy/(2*normxi^3*n


      for(j in (i+1):length(l)){
          xj = data[j,]
          normxj = sqr_norm[j,j]
          normxij=norm[i,j]

  partial_D_xixj = (xi*xj)/(normxi*normxj)-(xi*xi)*normxij/(2*normxi*normxj^3)-(xj*xj)*normxij/(2*normxi


             Add = Add + partial_D_xixj * (w2 * constraint2(i,j,d) + w6 * constraint6(i,j,d))
      }
    Add = Add + partial_D_xiy
    print(i)
    }
    #update the matrix

    Anew= A + eta * diag(Add)

 diff=eta*diag(Add)
  return(list(Y=Y,A=Anew,diff=diff))
}
```

EM algorithm completed - to be continued - pseudo-code

```
EM_algorithm = function(d, t=2, w2 = 0.7, w6 = w2^t ,tau=0.01, eta=0.01, max.iter=3){

  A=diag(1,dim(dtm_train_tfidf[[d]])[2],dim(dtm_train_tfidf[[d]])[2])
  p=dim(dtm_train_tfidf[[d]])[2]
  k=length(unique(author_id[[d]]))
  iter=0
  diff=A
  l=Initialization(A,d,k)
  Y=matrix(NA,k,p)
  for(j in 1:k){
    X = as.matrix(dtm_train_tfidf[[d]])
    Z=as.matrix(X[l==j,],ncol=dim(X)[2])#subset the Xi to the one with the label
```

```
    Y[j,] = col_sums(Z)
    Y[j,] = Y[j,]/(crossprod(Y[j,],A)%*% Y[j,])[1]
  }
   eta=eta*0.1
  while(iter<max.iter & norm(diff, type="1") > tau){

    l = Estep(A=A,Y=Y,l=l, d=d, w2 = 0.7, w6 = w2^t)


    # get the updated Y and A and changes between iterations
    Result = Mstep(A, l, Y, d, w2 = 0.7, w6 = w2^t, eta=eta)

    Y = Result[[1]]
    A = Result[[2]]
    diff=Result[[3]]
    iter=iter+1

  }
  return(list(l=l,A=A))
}
```

```
t=2
w2 = 0.7
w6 = w2^t
tau=0.01
eta=0.01
max.iter=3

A=diag(1,dim(dtm_train_tfidf[[d]])[2],dim(dtm_train_tfidf[[d]])[2])
  p=dim(dtm_train_tfidf[[d]])[2]
  k=length(unique(author_id[[d]]))
  iter=0
  diff=A
  l=Initialization(A,d,k)
  Y=matrix(NA,k,p)
  for(j in 1:k){
    X = as.matrix(dtm_train_tfidf[[d]])
    Z=as.matrix(X[l==j,],ncol=dim(X)[2])#subset the Xi to the one with the label
    Y[j,] = col_sums(Z)
    Y[j,] = Y[j,]/(crossprod(Y[j,],A)%*% Y[j,])[1]
  }
   eta=eta*0.1
```

Step 4: Clustering

```
label=list(1:length(data.files))
d=6
ss=Sys.time()
label[[d]]=EM_algorithm(d=d, w2 = 0.7, w6 = w2^2 ,tau=0.01, eta=0.1, max.iter=3)[[1]]
```

```
## [1] 1
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
```

```
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
## [1] 104
## [1] 105
## [1] 106
## [1] 107
## [1] 108
## [1] 109
```

```
## [1] 110
## [1] 111
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
```

```
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
## [1] 104
## [1] 105
## [1] 106
```

```
## [1] 107
## [1] 108
## [1] 109
## [1] 110
## [1] 111
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
```

```
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
```

```
## [1] 104
## [1] 105
## [1] 106
## [1] 107
## [1] 108
## [1] 109
## [1] 110
## [1] 111
```

```
Sys.time()-ss
```

```
## Time difference of 20.84981 secs
```

```
#!!Caution:If the above code can not run, pls first run the part before(while) in the EM algorithm.then
#here only the dataset 6 used for the test
#pretty quick for processing the data, only takes 22.99206s
```

Step 5: Evaluation

To evaluate the performance of the method, it is required to calculate the degree of agreement between a set of system-output partitions and a set of true partitions. In general, the agreement between two partitioins is measured for a pair of entities within partitions. The basic unit for which pair-wise agreement is assessed is a pair of entities (authors in our case) which belongs to one of the four cells in the following table (Kang et at.(2009)):

Matching matrix for the agreement between two sets of clusters

		Gold standard clusters (G)	
		Match	Mismatch
Machine-generated clusters (M)	Match	a	b
	Mismatch	c	d

Let M be the set of machine-generated clusters, and G the set of gold standard clusters. Then. in the table, for example, a is the number of pairs of entities that are assigned to the same cluster in each of M and G. Hence, a and d are interpreted as agreements, and b and c disagreements. When the table is considered as a confusion matrix for a two-class prediction problem, the standard "Precision", "Recall","F1", and "Accuracy" are defined as follows.

$$\text{Precision} = \frac{a}{a+b}$$
$$\text{Recall} = \frac{a}{a+c}$$
$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
$$\text{Accuracy} = \frac{a+d}{a+b+c+d}$$

```
library("gtools")
source('~/Documents/Github/Spr2017-proj4-team1/lib/evaluation_measures.R')

answer<-list(1:length(data.files))
d=6
```

```
   author_id_sep<-author_id[[d]]
  l_sep<-label[[d]]
  evaluation<-function(l_sep,author_id_sep){
    n <- length(l_sep)
    author_id_mat <- matrix(NA,n,n)
    for(i in 1:n){
      author_id_mat[i,] <- as.numeric(sapply(author_id_sep,"==",author_id_sep[i]))
    }

   l_mat <- matrix(NA,n,n)

    for(j in 1:n){
      l_mat[j,] <- sapply(l_sep,"==",l_sep[j])*3+2
    }

    match_matrix <- author_id_mat +l_mat


    mis.mis <- sum(match_matrix==2)/2
mat.mis <- sum(match_matrix==3)/2

    mis.mat <- sum(match_matrix==5)/2

    mat.mat <- (sum(match_matrix==6)-n)/2

    pre<-mat.mat/(mat.mat+mat.mis)
    recal<-mat.mat/(mat.mat+mis.mat)
    F1<-2*pre*recal/(pre+recal)
    accur<-(mis.mis+mat.mat)/sum(mis.mis,mis.mat,mat.mis,mat.mat)
    result<-c(pre,recal,F1,accur)
    return(list(pre=result[1],recal=result[2],F1=result[3],accur=result[4]))
    }

  answer[[d]]<-evaluation(l_sep, author_id_sep)

answer[[d]]
```

```
## $pre
## [1] 0.6003289
##
## $recal
## [1] 0.3571429
##
## $F1
## [1] 0.4478528
##
## $accur
## [1] 0.8552124
```

```
#answer[[d]] will be a list precision ,recall,f1,accuracy
#Accuracy is 85%. So HMRF_EM is very efficient
```