paper5

# Step 0: Load the packages, specify directories

#setwd("........")

## Step 1: Load and process the data

# final version of data processing  
#similar with NB\_CO  
# do not paste here, using same data\_list  
# then, the function help us get the data.csv for further analysis  
# 3 5 8 13 14  
data\_gather=function(){  
hh=1:14  
num.paper<-length(data\_list[[hh]])  
  
auther.id<-NULL  
coauther<-NULL  
for(i in 1:num.paper){  
 auther.id[i]<-data\_list[[hh]][[i]][[1]]  
 coauther<-unique(c(coauther,data\_list[[hh]][[i]][[3]]))  
}  
  
df<-data.frame(auther.id)  
  
num.coauther<-length(coauther)  
  
df<-cbind(auther.id,id\_co)   
  
get.coauther<-function(i){  
 coauthers<-rep(0,num.coauther)  
 for(j in 1:length(data\_list[[hh]][[i]][[3]])){  
 coauthers[which(data\_list[[hh]][[i]][[3]][j]==coauther)]<- 1  
}  
return(coauthers)   
}  
id\_co<-NULL  
for(i in 1:num.paper){  
 id\_co<-rbind(id\_co,get.coauther(i))  
}  
colnames(id\_co)<-coauther  
df<-cbind(auther.id,id\_co)   
#write.csv(df,....)  
}

## Step 2: Clusterwise Scoring Function & Error-driven Online Training & Ranking MIRA

# similar to cost function, but we want to maximize this  
# the score would represent the accuracy of the training we get  
# each iteration would tell a better result, since it is greedy algorithm and focused on mistaken part  
sf=function(x,y){  
 sum=0  
 for(i in unique(y)){  
 sum=sum+ sum(x==i)/length(x)# wrong?  
 }  
 return (sum)  
}  
  
# runtime  
start.time <- Sys.time()  
# run the algorithm which would predict the authorID class  
end.time <- Sys.time()  
time\_sclust <- end.time - start.time  
time\_sclust

## Paper5 C/E/M

#final function  
update.weights=function(data,weights,target)  
{  
pred=which.max(weights%\*%t(data.matrix(data)))  
tau=(t(data.matrix(weights[pred,]-weights[target,]))%\*%t(data.matrix(data))+1)/(2\*sum(data^2))  
if(pred!=target)  
{  
weights[pred,]=as.vector(weights[pred,])-as.vector(unlist(min(tau[1,1],0.008)\*data))  
weights[target,]=as.vector(weights[target,])+as.vector(unlist(min(tau[1,1],0.008)\*data))  
   
   
}  
   
return(weights)   
   
   
}  
  
predict.mira=function(data,weights)  
{  
 return(apply(weights%\*%t(data.matrix(data)),2,which.max))   
}  
  
mira=function(x,y,levels=length(unique(y)))  
{  
#y=as.numeric(as.factor(y))  
weights=matrix(rep(1,levels\*length(x[1,])),nrow=levels)   
weights=update.weights(x[1,],weights,y[1])  
pred=predict.mira(x,weights)  
errorid=which(pred!=y)  
diff=1  
while(diff>0 && (length(errorid)!=0))  
{  
weights=update.weights(x[errorid[1],],weights,y[errorid[1]])  
pred=predict.mira(x,weights)  
errorid2=which(pred!=y)  
diff=length(errorid)-length(errorid2)  
errorid=errorid2  
}  
return(weights)   
}

## Evaluation

Calculate the accrucy of the 14 authors. Need to mention, the error-driven is based on selecting first error, so this is quite random. The result may vary on the selection of training & testing and selecting. And some of the data set is too small, so splitting on multi-classification would sometimes has too many sample of one kind only in train/test.

run=function(){  
 res=c()  
 #set wd to get dfs  
  
 temp=c("df1.csv","df2.csv","df3.csv","df4.csv","df5.csv","df6.csv","df7.csv","df8.csv","df9.csv","df10.csv","df11.csv","df12.csv","df13.csv","df14.csv")  
 for (i in 1:length(temp)) {  
 df = read.csv(temp[i], header = TRUE)  
 x=df[,-c(1,2)]  
 y=as.numeric(as.factor(df$auther.id))  
 trainid=sample(1:length(y),length(y)/2)  
 trainx=x[trainid,]  
 trainy=y[trainid]  
 testx=x[-trainid,]  
 testy=y[-trainid]  
 weights=mira(trainx,trainy,levels=length(unique(y)))  
 o=predict.mira(testx,weights)  
 res[i]=sum(testy==o)/length(testy)  
 cat(i)  
 cat(' ')  
 }  
 return (res)  
}  
run()

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14

## [1] 0.62283737 0.80327869 0.28678304 0.88586957 0.14366197 0.82142857  
## [7] 0.82558140 0.35344828 0.87142857 0.66233766 0.83076923 0.86893204  
## [13] 0.09836066 0.42812006