Data Science Machine

Hamza Zafar

November 6, 2015

For validting the machine KDD-2014 data is considere.The Data Science Machine mainly consists two parts.

1. Deep Feature Synthesis
2. Machine Learning pipline using Gaussian Copla Process

## Deep Feature Synthesis:

Since data of KDD-2014 is large enough to manipulate by R, so mysql is connected as backhand to handle the data by creating queries. The data from KDD-2014 was stored in 5 different files, it is manually converted into the relational data base detailed in the proposed research paper, that resulted in 9 data tables, the subset of them are considered for making features.

In Section 1 there is snippet code, which asks the user about the Entity, Related Entity, and features of related entity to be added in the Entity, and also apply defined function to the selected feature. Calling the *make\_feature* connects through the database, and lists all available datatable to considered for. Then we can select the relevent tables 'Entity' & 'Related Entity' from the list. Then function asks for the features from the related Entity to be added as direct or backward. In case of case of backward, it also provide the user information about the type of feature (text, double) to select the relevant Function as 'sum, count, avg, std'. Then finally combining 'rfeat, dfeat' it applies function (mode) for making efeat features. It iterates through the data untill user want.

The all data is sotred in databases to lessen the work for R.

## Section #1:

require(dplyr)  
require(RMySQL)  
  
user <- 'root'  
pass <- c()  
dbname <- 'data\_science\_machine'  
host <- '127.0.0.1'

make\_features <- function(Ek = '', El = '', Fun ='count'){  
 qry <- c()  
 entityLoop <- 1  
 log <- data.frame( "Entity" = character(), "Related\_Entity" = character(),  
 "Related\_Feature" = character(), "Relation" = integer(),  
 "Fun" = character(), "Query" = character(),  
 stringsAsFactors=FALSE)   
   
 #'  
 #'iterate to select multiple entities  
 #'  
   
 while(entityLoop == 1){  
 featureLoop <- 1;  
   
 print(dbListTables(mydb))  
   
 # print("Used Features: ")  
   
   
 Ek <- readline(prompt = "select the feature Entity:\t")  
   
 print (paste('Related Entitis of', Ek, ':'))  
 print(unique(log[log$Related\_Entity==El]$Related\_Entity))  
   
 El <- readline(prompt = "select the relational Entity:\t")  
 # Ek <- 'vendor\_ety'  
 # El <- 'resource\_ety'  
   
 relation <- readline(prompt = 'Forward (1) & backward (0):\t')  
 # relation <- 0  
   
 #'  
 #' iterate to select multiple features  
 #'  
   
 while(featureLoop == 1){  
   
 print(dbListFields(mydb, El))  
 print("Related Entities Used Before:")  
 print( log[log$Entity==Ek & log$Related\_Entity==El]$Related\_Feature)  
   
 relatedFeature <- readline('select feature:\t')  
   
 k <- readline(prompt = 'join key: \t')  
   
 # k <- 'vendorid'  
   
 if( relation == '1'){  
   
 Fun <- ''  
 qry <- dfeat(Ek = Ek, El = El,  
 relatedFeature = relatedFeature, k = k)  
   
# qry <- paste('select ', Ek ,'.\*, ', El, '.', relatedFeature, ' from ', Ek,   
# ' left join ', El, ' on ',   
# Ek, '.', k, '=', El, '.', k, ';' , sep ='')  
#   
 }  
 if (relation == '0'){  
   
 print(paste(  
 'Type of the feautre:', get\_feature\_type(  
 El, relatedFeature  
 )  
 ))  
   
 Fun <- readline('sum, avg, std, count: ? \t')  
 # log$Fun <- Fun  
   
 qry <- rfeat(Ek = Ek, El = El, Fun = Fun,  
 relatedFeature = relatedFeature, k = k)   
   
# qry <- paste('select ', Ek ,'.\*, ', Fun, '(', El, '.', relatedFeature, ') as ',  
# Fun, '\_', relatedFeature, ' from ', Ek, ' left join ', El, ' on ',   
# Ek, '.', k, '=', El, '.', k, ' group by ', Ek, '.', k, ';' , sep ='')  
   
 }  
   
 insert\_feature\_db(qry, Ek)  
   
 featureLoop <- readline(prompt = 'More Features (yes=1, No=0):\t')  
   
 log[nrow(log)+1,] <- c(Ek, El, relatedFeature,   
 relation, Fun, qry)  
 #print(log)  
 }  
 efeatLoop <- readline(prompt = 'EFEAT (yes=1, No=0):\t')  
 if(efeatLoop ==1){  
 nam <- paste(Ek, 'features', sep = '\_')  
 for (i in dbListFields(mydb, nam)){  
 efeat(nam, i)  
 }  
 }  
 entityLoop <- readline(prompt = 'More Entities (yes=1, No=0):\t')  
 }  
}

dfeat <- function(Ek, El, relatedFeature, k){  
 paste('select ', Ek ,'.\*, ', El, '.', relatedFeature, ' from ', Ek,   
 ' left join ', El, ' on ',   
 Ek, '.', k, '=', El, '.', k, ';' , sep ='')  
   
}  
  
rfeat <- function(Ek, Fun, El, relatedFeature, k){  
 paste('select ', Ek ,'.\*, ', Fun, '(', El, '.', relatedFeature, ') as ',  
 Fun, '\_', relatedFeature, ' from ', Ek, ' left join ', El, ' on ',   
 Ek, '.', k, '=', El, '.', k, ' group by ', Ek, '.', k, ';' , sep ='')  
}  
  
efeat <-function(Ek,relatedFeature){  
   
# Ek <- 'test'  
# relatedFeature <- 'count\_resource\_type'  
   
 countQry <- paste0('select ', relatedFeature, ', count(', relatedFeature,  
 ') as a from ', Ek, ' group by ', relatedFeature, ';')  
   
   
 rs <- dbSendQuery(mydb, countQry)  
 count <- fetch(rs,n=-1)  
   
   
 repVal <- filter(count, a==max(a))[[1]]  
   
 # print(length(repVal))  
 # print(repVal)  
 # print(repVal[1])  
 #   
 # print(class(repVal[1]))  
 #   
 # print(repVal[1]== 'numeric')  
 #   
 # if(repVal[1]== 'numeric')  
 # {  
 # updQry <- paste0('update ', Ek, ' set ', relatedFeature, ' = ', repVal[1],  
 # ' where ', relatedFeature, ' is null ;')  
 # }  
   
   
 updQry <- paste0("update ", Ek, " set ", relatedFeature, " = '", repVal[1],  
 "' where ", relatedFeature, " is null ;")  
   
 # print(updQry)  
 # readline()  
 dbSendQuery(mydb, updQry)  
 print(paste(Ek,'feature', relatedFeature, 'Updated'))  
   
}

insert\_feature\_db <- function(qry, Ek){  
   
 nam <- paste(Ek, 'features', sep = '\_')  
   
 if (is.element(nam, dbListTables(mydb))){  
 createQry <- paste0('create table temp as ', qry)  
 dbSendQuery(mydb,createQry)  
   
 k <- dbListFields(mydb, nam)[1]  
 col <- dbListFields(mydb, 'temp')  
 col <-col[length(col)]  
   
 joinQry <- paste('select ', nam ,'.\*, ', 'temp.', col, ' from ', nam,   
 ' left join temp on ',   
 nam, '.', k, '=', 'temp.', k, ';' , sep ='')  
   
 rs <- dbSendQuery(mydb, joinQry)  
 temp <- fetch(rs,n=-1)  
   
 dbWriteTable(mydb,name = nam, value = temp,  
 overwrite = T, row.names = F)  
   
 rm(temp)  
 dbSendQuery(mydb,'Drop table temp')  
   
 print('feature added')  
 }   
 else{  
 createQry <- paste0('create table ', nam, ' as ', qry)  
 dbSendQuery(mydb,createQry)  
 print('feature created')  
   
 }  
   
}  
  
get\_feature\_type <- function(entity, feature=''){  
 feature\_info <- dbSendQuery(mydb, paste0('DESCRIBE ', entity, ';'))  
 db\_info <- fetch(feature\_info, n=-1)  
 # db\_info  
   
 db\_info[db\_info$Field==feature,]$Type  
   
}