NB-paper&journal

Vic Jiang

2017.4.9

Step 0: Load the packages, specify directories

```
#setwd("your path here/lib")
if (!require("stringr")) install.packages("stringr")
```

```
## Loading required package: stringr
```

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

```
## Loading required package: tm
```

```
## Loading required package: NLP
```

```
library(stringr)
library(tm)
```

Step 1: Load and process the data

```
setwd("F:/statistics material/2017Spring/5243 applied data science/Spr2017-proj4-team-5/lib")
data.lib="../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)

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

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

  # remove unwanted characters
  char_notallowed <- "\\@#$%^&?" # characters to be removed
  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,
       paper_id,
       coauthor_list,
       paper_title,
       journal_name)
}

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

for(i in 1:length(data.files)){
  ## Step 0 scan in one line at a time.

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

}
```

Step 2: Build function to calculate the accuracy of classification.

Some auxiliary function to be used in main function "classify"

```
#Write a function that use to transform the journal/paper title into seperate words by lower case, remo
cleaning2 <- function(list,element=5){
  document <- list[[element]]
  document <- Corpus(VectorSource(document))
  document = tm_map(document, content_transformer(tolower))
  document = tm_map(document, removePunctuation)
  document = tm_map(document, removeWords, stopwords("english"))
  document <- document[[1]]$content
  document <- strsplit(document,split = " +")[[1]]
  list[[element]] <- document
  return(list)
}
cleaning1 <- function(list1,element=5){
  list1 <- lapply(list1,cleaning2,element=element)
  return(list1)
}

#A function that get the word of journal/paper title of each person
word_per <- function(name,number,element=5,data=data_train){
  jour_train_word <- vector("list",(length(number)-1))
  for(i in 1:(length(number)-1)){
    for(j in (number[i]+1):number[1+i]){
      jour_train_word[[i]] <- c(jour_train_word[[i]],data[[name]][[j]][[element]])
    }
  }
  return(jour_train_word)
}
```

Define the main function "classify".

```
#list_amname is the data set(train and test included). If element=4, then we are doing classification a
classify <- function(list_amname=data_list,element=5,seed=1){
  #Set seed
  set.seed(seed)
  #Get the data list that seperate the journal/paper title into single words
  data_list_clean <- lapply(list_amname,cleaning1,element=element)
  #Get the number of each name
  n <- length(data_list_clean)
```

```
#Get the number of each name's article.
n_total <- c()
for(i in 1:n){
  n_total[i] <- length(data_list_clean[[i]])
}

#Calculate how many different people in each name
number <- vector("list",n)
m <- c()
for(i in 1:n){
  for(j in 1:n_total[i]){
    number[[i]][j] <- data_list_clean[[i]][[j]][[1]]
  }
  m[i] <- max(number[[i]])
}

#Get the training set and test set.
placeofper <- c()
trainofper <- c()
testofper <- c()
data_train <- vector("list",n)
data_test <- vector("list",n)
numofper <- vector("list",n)
for(i in 1:n){
  numofper[[i]] <- table(number[[i]])
  numofper[[i]] <- numofper[[i]][order(match(names(numofper[[i]]),unique(number[[i]])))]
  for(k in 1:length(numofper[[i]])){
    placeofper[k] <- sum(numofper[[i]][1:k])
  }
  names(placeofper) <- names(numofper[[i]])
  placeofper <- c(0,placeofper)
  for(j in 1:(length(placeofper)-1)){
    sampleper <- sample((placeofper[j]+1):placeofper[j+1],ceiling(numofper[[i]][j]/2))
    trainofper <- c(trainofper,sampleper)
  }
  data_train[[i]] <- data_list_clean[[i]][trainofper]
  data_test[[i]] <- data_list_clean[[i]][-trainofper]
  trainofper <- c()
  placeofper <- c()
}

#Get the number of each name's article.
n_train <- c()
n_test <- c()
for(i in 1:n){
  n_train[i] <- length(data_train[[i]])
  n_test[i] <- length(data_test[[i]])
}

#Get the specific number list of each person's article in data
train_numofper <- vector("list",n)
train_placeofper <- vector("list",n)
for(i in 1:n){
```

```r
    train_numofper[[i]] <- ceiling(numofper[[i]]/2)
    for(k in 1:length(train_numofper[[i]])){
      train_placeofper[[i]][k] <- sum(train_numofper[[i]][1:k])
    }
    train_placeofper[[i]] <- c(0,train_placeofper[[i]])
  }

  #Get the word of journal title of each person in each name
  jour_train_word <- vector("list",n)
  for(i in 1:n){
    jour_train_word[[i]] <- word_per(name=i,number = train_placeofper[[i]],element=element,data = data_
  }

  #Compute each word's times of each person in each name
  jour_train_time <- jour_train_word
  for(i in 1:n){
    for(j in 1:length(jour_train_word[[i]])){
      jour_train_time[[i]][[j]] <- table(jour_train_word[[i]][[j]])
      jour_train_time[[i]][[j]] <- jour_train_time[[i]][[j]][names(jour_train_time[[i]][[j]])!=""]
    }
  }

#As every article have a journal title, P(CO|Xi) is always 1, and thus P(N|Xi) is 0.
#Total number of words of each name.
  num_word_name <- rep(0,n)
  for(i in 1:n){
    for(j in 1:length(jour_train_word[[i]])){
      num_word_name[i] <- num_word_name[i]+length(jour_train_word[[i]][[j]])
    }
  }

  #P(Seen|CO,Xi), P(Unseen|CO,Xi), P(A3k|Seen,CO,Xi) and P(A3k|Unseen,CO,Xi)
  for(i in 1:n){
    for(j in 1:length(jour_train_time[[i]])){
      prob_seen <- sum(jour_train_time[[i]][[j]]>=2)/length(jour_train_time[[i]][[j]])
      jour_train_time[[i]][[j]] <- c(prob_seen,(1-prob_seen),jour_train_time[[i]][[j]])
      names(jour_train_time[[i]][[j]])[1:2] <- c("prob_seen","prob_unseen")

      #Seperate the probability of seen, prob of unseen, seen words and unseen words into 4 element.
      transition <- jour_train_time[[i]][[j]]
      jour_train_time[[i]][[j]] <- vector("list",3)
      jour_train_time[[i]][[j]][[1]] <- transition[1] #P(Seen|CO,Xi)
      jour_train_time[[i]][[j]][[2]] <- transition[2] #P(Unseen|CO,Xi)
      jour_train_time[[i]][[j]][[3]] <- data.frame(transition[c(-1,-2)])
      names(jour_train_time[[i]][[j]]) <- c("prob_seen","prob_unseen","word")
      names(jour_train_time[[i]][[j]][[3]]) <- c("times")
      jour_train_time[[i]][[j]][[3]]$seen <- jour_train_time[[i]][[j]][[3]]$times/sum(jour_train_time[[
      unseen <- 1/(num_word_name[i]-length(jour_train_time[[i]][[j]][[3]]$times))
      jour_train_time[[i]][[j]][[3]]$unseen <- rep(unseen,nrow(jour_train_time[[i]][[j]][[3]])) #P(A3k|
    }
  }

  for(i in 1:n){
    for(j_test in 1:n_test[i]){
      prob <- c()
      for(j_train in 1:length(jour_train_time[[i]])){
        word_match <- match(data_test[[i]][[j_test]][[element]],rownames(jour_train_time[[i]][[j_train]]
        if(sum(is.na(word_match))!=0){
          word_match[is.na(word_match)] <- (nrow(jour_train_time[[i]][[j_train]]$word)+10)
        }
        word_seen_prob <- jour_train_time[[i]][[j_train]]$word$seen[word_match]
        if(sum(is.na(word_seen_prob))!=0){
          word_seen_prob[is.na(word_seen_prob)] <- 0
        }
        inter <- word_seen_prob*jour_train_time[[i]][[j_train]][[1]]+rep(jour_train_time[[i]][[j_train]]
        prob[j_train] <- sum(log(inter))+log(numofper[[i]][j_train]/sum(numofper[[i]]))
      }
      data_test[[i]][[j_test]][[1]] <- c(data_test[[i]][[j_test]][[1]],names(numofper[[i]])[which.max(p
    }
  }

  wrong <- rep(0,n)
  count <- rep(0,n)
  for(i in 1:n){
    for(j in 1:n_test[i]){
      wrong[i] <- wrong[i]+(data_test[[i]][[j]][[1]][1]!=data_test[[i]][[j]][[1]][2])
      count[i] <- count[i]+1
    }
  }
  accurarcy <- (rep(1,length(wrong))-wrong/count)
  return(accurarcy)
}
```

Step 3: get the performance

```
seed_set <- sample(1:10000,10)
accurarcy_jour <- matrix(NA,nrow = 12,ncol = 14)
accurarcy_paper <- matrix(NA,nrow = 12,ncol = 14)
counting <- 1
ptm <- proc.time() #Start the clock
for(i in seed_set){
  accurarcy_jour[counting,] <- classify(seed = seed_set[counting])
  counting <- counting+1
}
time_take_jour <- proc.time()-ptm #End the clock for journal title
ptm <- proc.time()
counting <- 1
for(i in seed_set){
  accurarcy_paper[counting,] <- classify(element = 4,seed = seed_set[counting])
  counting <- counting+1
}
time_take_paper <- proc.time()-ptm
accurarcy_jour[11,] <- apply(accurarcy_jour[1:10,],2,mean)
accurarcy_jour[12,] <- apply(accurarcy_jour[1:10,],2,sd)
```

```
rownames(accurarcy_jour) <- c(1:10,"mean","sd")
accurarcy_paper[11,] <- apply(accurarcy_paper[1:10,],2,mean)
accurarcy_paper[12,] <- apply(accurarcy_paper[1:10,],2,sd)
rownames(accurarcy_paper) <- c(1:10,"mean","sd")

accurarcy_jour[11,]#mean for journal title
```

```
##  [1] 0.5551601 0.6900000 0.4507772 0.6348066 0.4740795 0.5259259 0.6903614
##  [8] 0.7326039 0.7525547 0.6684932 0.7314961 0.8440594 0.5058989 0.5272876
```

```
accurarcy_jour[12,]#sd for journal title
```

```
##  [1] 0.01941958 0.03824870 0.02061716 0.03469390 0.02025833 0.04294451
##  [7] 0.02605804 0.01594023 0.03713154 0.04370223 0.01908987 0.02382761
## [13] 0.01691624 0.02260250
```

```
accurarcy_paper[11,]#mean for paper title
```

```
##  [1] 0.7188612 0.7250000 0.6406736 0.8082873 0.6580265 0.4907407 0.6771084
##  [8] 0.7722101 0.8437956 0.7684932 0.7299213 0.8608911 0.6672753 0.6831699
```

```
accurarcy_paper[12,]#sd for paper title
```

```
##  [1] 0.024368492 0.028054180 0.018563481 0.017673802 0.009283414
##  [6] 0.041177976 0.027707933 0.009971654 0.021816556 0.030597015
## [11] 0.024635591 0.018144313 0.021315127 0.013917161
```