Project 4 - Paper 5

Team 8 - Yijia Pan, Xiaowo Sun

4/14/2017

In this notebook, we focus on the method mentioned in the paper written by Song, Yang, et al. for the name disambiguation issue. We adopted Clusterwise Score Function, and used Error-Driven Online Training to generate examples and Ranking MIRA as the loss function.

Step 0: Load the packages, specify directories

```
#setwd("~/Project4_WhoIsWho/doc")
if (!require("pacman")) install.packages("pacman")
pacman::p_load(text2vec, dplyr, qlcMatrix, kernlab, knitr)
library(stringr)
library(qlcMatrix)
```

Step 1: Data & Feature Preparation

The features used for our experiments are as follows. We use the attributes of co-authors, paper titles and journals provided in the nameset folder. After comparing different similarity measures, we use TF-IDF measure to calculate similarity. Before that, we construct a document-term matrix (DTM). Here we use the data processed by our teammates.

```
load("../output/dtm_list.RData")
load("../output/tfidf_list.RData")
dtm_train_tfidf<-tfidf_list[[1]]$tfidf
label<-rownames(as.data.frame(as.matrix(dtm_list[[1]]$dtm)))
```

Step 2: Algorithm

Clustering

We use agglomerative clustering method.

```
lamda<-vector(length=dim(dtm_train_tfidf)[2])

lamda<-rep(1,dim(dtm_train_tfidf)[2])

L<- rep(lamda,each=dim(dtm_train_tfidf)[1])
L<- matrix(L,nrow = dim(dtm_train_tfidf)[1], ncol = dim(dtm_train_tfidf)[2])

dtm_train_tfidf<-dtm_train_tfidf*L
docsdissim <- cosSparse(t(dtm_train_tfidf))
rownames(docsdissim) <- c(1:nrow(dtm_train_tfidf))
colnames(docsdissim) <- c(1:nrow(dtm_train_tfidf))

h <- hclust(as.dist(docsdissim), method = "ward.D")
```

Clusterwise Scoring Function

We define the clusterwise scoring function as the sum of scores for each cluster.

```
s<-NULL
Score<-function(lamda,result_hclust){
  f<-matrix(NA,nrow=dim(dtm_train_tfidf)[2],ncol=length(unique(result_hclust)))
  for (i in 1:length(unique(result_hclust))){

       cluster<-unique(result_hclust)
       num<-count[result_hclust == cluster[i]]

       if (length(num) > 1){
       f[,i]<-colMeans(dtm_train_tfidf[num,])
       }
       else{
       f[,i]<-dtm_train_tfidf[num,]
       }

       s[i]<-t(lamda) %*% f[,i]
   }
  S<-sum(s)
  return(S)
}
```

CreateExamplesFromErrors by Error-driven Online Training

We employ a sampling scheme that selects training examples based on errors that occur during inference on the labeled training data (Culotta, Aron, et al., 2007).

```
source('../lib/evaluation_measures.R')
count<-1:length(label)
M<-length(label)
K<-(M-1):2
acc<-NULL
sco<-NULL
neighbor<-matrix(NA, nrow = length(label),ncol = 100)
best<-matrix(NA, nrow = length(label), ncol = 2)

errordriven<-function(h){
for (m in 1:(M-2)){
   k<-K[m]
     result_hclust <- cutree(h,k)
     for (i in 1:k){
       num<-count[result_hclust == i]
       if (length(unique(label[num]))>1)
        {
         for (j in 1:100){
         neighbor[,j]<-sample(1:k,length(label),replace  = T)
         matching_matrix_hclust <- matching_matrix(label,neighbor[,j])
         acc[j] <- performance_statistics(matching_matrix_hclust)[[1]]
         sco[j]<-Score(lamda,neighbor[,j])
         }
       }
```

```
    }
       best[,1]<-neighbor[,which.max(acc)]
       best[,2]<-neighbor[,which.max(sco)]
 }
  return(best)
}
```

UpdateParameters by Ranking MIRA

We use a variant of MIRA (Mar- gin Infused Relaxed Algorithm), a relaxed, online max- imum margin training algorithm (Crammer & Singer 2003). As discussed in the paper, We updates the parameter vector with three constraints: (1) the better neighbor must have a higher score by a given margin, (2) the change to should be minimal, and (3) the inferior neighbor must have a score below a user-defined threshold (Culotta, Aron, et al., 2007).

```
s1<-NULL
s2<-NULL

UpdateMatrix<-function(result_hclust){

  for (i in 1:length(unique(result_hclust))){

       cluster<-unique(result_hclust)
       num<-count[result_hclust == cluster[i]]

       if (length(num) > 1){
       f[,i]<-colMeans(dtm_train_tfidf[num,],na.rm = T)
       }
       else{
       f[,i]<-dtm_train_tfidf[num,]
       }
   }
       ff<-colMeans(t(f),na.rm = T)
       return(ff)
}


fr <- function(x){
  sum((x-lamda)^2)
}
```

Whole Algorithm

```
lamda<-rep(1,dim(dtm_train_tfidf)[2])
margin<-0.01
tao<--30

LAMDA <- matrix (NA,nrow = length(lamda),ncol = 2)
LAMDA[,1] <- lamda

for (p in 1:100){
```

```
# The feature matrix after weighed by parameters
L<- rep(LAMDA[p],each=dim(dtm_train_tfidf)[1])
L<- matrix(L,nrow = dim(dtm_train_tfidf)[1], ncol = dim(dtm_train_tfidf)[2])
dtm_train_tfidf<-dtm_train_tfidf*L
docsdissim <- cosSparse(t(dtm_train_tfidf))
rownames(docsdissim) <- c(1:nrow(dtm_train_tfidf))
colnames(docsdissim) <- c(1:nrow(dtm_train_tfidf))

# Cluster based on the feature matrix
h <- hclust(as.dist(docsdissim), method = "ward.D")

# Drive the error
best<-errordriven(h)

# Update the parameters
f1<-UpdateMatrix(best[,1])
f2<-UpdateMatrix(best[,2])
f3<-f1-f2
u1<-rbind(f3,-f2)
c1<-c(margin,tao)

fr <- function(x){
  sum((x-LAMDA[p])^2)
}

optim<-constrOptim(LAMDA[p],fr,grad = NULL, ui = u1,ci = c1)
lamda<-optim$par
LAMDA<-cbind(LAMDA,lamda)

# Converage
if (sum((LAMDA[p+1]-LAMDA[p])^2)<0.1)
{
  break
}
}
```

Result Cluster

```
lamdaoptim<-LAMDA[,dim(LAMDA)[2]]
L<- rep(lamdaoptim,each=dim(dtm_train_tfidf)[1])
L<- matrix(L,nrow = dim(dtm_train_tfidf)[1], ncol = dim(dtm_train_tfidf)[2])

dtm_train_tfidf<-dtm_train_tfidf*L
docsdissim <- cosSparse(t(dtm_train_tfidf))
rownames(docsdissim) <- c(1:nrow(dtm_train_tfidf))
colnames(docsdissim) <- c(1:nrow(dtm_train_tfidf))

h <- hclust(as.dist(docsdissim), method = "ward.D")

result_hclust <- cutree(h,k)
```

Other data we use

```
# After we calculate the parameters, we can employ it on the other 9 files and can average the paramter
```

Step 3: 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))

For evaluation, we use the standard "Precision", "Recall","F1", and "Accuracy" defined as follows.

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

```
source('../lib/evaluation_measures.R')
matching_matrix_hclust <- matching_matrix(label,result_hclust)
performance_hclust <- performance_statistics(matching_matrix_hclust)
df <- data.frame(precision=performance_hclust$precision,
recall=performance_hclust$recall,
f1=performance_hclust$f1,
accuracy=performance_hclust$accuracy,
time=c(time_hclust))
```