Project 4 - Paper 4

Ken Chew, Terry Li, Yue Jin

04/12/2017

This file is currently a template for implementing one of the suggested papers, Han, Zha, & Giles (2005). Due to the nature of the method, we only implement the method on a subset of the data, "AKumar.txt". In your project, you need to work on the whole dataset. You should follow the same structure as in this tutorial, but update it according to the papers you are assigned.

Step 0: Load the packages, specify directories

```
if (!require("pacman")) install.packages("pacman")
pacman::p_load(text2vec, dplyr, qlcMatrix, kernlab, knitr)
library(stringr)
source("../lib/cleaning.R")

setwd("~/Dropbox/Project4_WhoIsWho/doc")
# 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

For each record in the dataset, there are some information we want to extract and store them in a regular form: canonical author id, coauthors, paper title, publication venue title.

```
data.lib="../data/nameset"
data.files=list.files(path=data.lib, "*.txt")

data.files

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

query.list
```

Step 2: Feature design

Since we do not have first page of the papers, we adapted paper 4 and used paper titles, coauthors and journal names to design features for citations.

First let's create a vocabulary-based term matrix. we do so in two ways (to compare performance and robustness). The first way: Create a DTM using each author as the corpus. i.e. AGupta, AKumar. 14 DTM's in total.

```
## generate DTM/tfidf for 14 files individually

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])
}

dtm_list <- list()
for (file in 1:length(data_list)) {
  a2 <- list()
  for (i in 1:length(data_list[[file]])) {
    a2 <- rbind(a2, data_list[[file]][[i]])
  }

  a2 <- data.frame(a2)
  a2 <- sapply(a2, unlist)
  a2 <- data.frame(a2)
  a2$combined <- as.character(a2$combined)

  it_train2 <- #itoken(paste(a2$paper_title, a2$coauthor_list, a2$journal_name),
               itoken(a2$combined,
               preprocessor = tolower,
               tokenizer = word_tokenizer,
               ids = a2$unique_paper_id,
               # turn off progressbar because it won't look nice in rmd
               progressbar = FALSE)
  vocab2 <- create_vocabulary(it_train2, stopwords = c("a", "an", "the", "in", "on",
                                                       "at", "of", "above", "under"))

  vectorizer <- vocab_vectorizer(vocab2)
  dtm_train <- create_dtm(it_train2, vectorizer)

  dtm_i <- list(name = query.list[file], dtm = dtm_train)
  dtm_list[[file]]  <- dtm_i
}

save(dtm_list, file = "../output/dtm_list.RData")
```

The second way: Create one DTM for all authors combined. one giant corpus potentially more robust considering each documents is rather small in size.

```
## generate DTM/tfidf for all 14 files combined as one

data_list = list()
dat = list()

data_list2 = 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_list2[[i]]=lapply(dat, f.line.proc, nam.query=query.list[i])
  data_list <- c(data_list, data_list2[[i]])
}


dtm_list_combined <- list()
a2 <- list()
for (line in 1:length(data_list)) {
  a2 <- rbind(a2, data_list[[line]])
  }

a2 <- data.frame(a2)
a2 <- sapply(a2, unlist)
a2 <- data.frame(a2)
a2$combined <- as.character(a2$combined)

it_train2 <- itoken(a2$combined,
                    preprocessor = tolower,
                    tokenizer = word_tokenizer,
                    ids = a2$unique_paper_id,
                    # turn off progressbar because it won't look nice in rmd
                    progressbar = FALSE)
vocab2 <- create_vocabulary(it_train2, stopwords = c("a", "an", "the", "in", "on",
                                                     "at", "of", "above", "under"))

vectorizer <- vocab_vectorizer(vocab2)
dtm_list_combined <- create_dtm(it_train2, vectorizer)

save(dtm_list_combined, file = "../output/dtm_list_combined.RData")
```

Perform Latent Dirichret Allocation on the DTM's forming k number of latent topics.

- Generate Theta - a probability distribution for each document over k topics.
- Theta corresponds to the probabilities associated with each topic assignment for each document
- Theta - number of rows= number of topics, number of columns= k (topics)
- Each row is a probability distribution that adds to 1.
- So each element in the matrix represent the probability/proportion that given document associate with the specific topic

```
library("topicmodels")
load("../output/dtm_list.RData")


# LDA parameters
burnin <- 4000
iter <- 1000
thin <- 200
seed <-list(2003,2,68,100001,765)
nstart <- 5
best <- TRUE
delta=0.1
```

```
# number of topics
k <- 10
alpha<-50/k


# run lda on each name
theta_list<-list(name=NULL,theta=NULL)
phi_list<-list(name=NULL,phi=NULL)

for (i in 1:length(query.list)){
  dtm<-dtm_list[[i]][[2]]
  rowTotals <- apply(dtm , 1, sum)
  dtm  <- dtm[rowTotals> 0, ]
  # Run LDA using Gibbs sampling
  ldaOut <-LDA(dtm, k, method="Gibbs",  control=list(nstart=nstart,
                                                     seed = seed,
                                                     best=best,
                                                     burnin = burnin,
                                                     alpha=alpha,
                                                     delta=delta,
                                                     iter=iter,
                                                     thin=thin))
  # theta_dj;
  # Probabilities associated with each topic assignment for each record
  theta <- as.data.frame(ldaOut@gamma)
  # phi_mj
  # the probability of using word m in topic j
  phi <- posterior(ldaOut)$terms

  temp1<-list(name=query.list[i], theta=theta)
  theta_list[[i]]<- temp1
  temp2<-list(query.list[i], phi)
  phi_list[[i]] <- temp2
}

## rerun LDA on combined dtm
load("../output/dtm_list_combined.RData")
dtm_combined<-dtm_list_combined
rowTotals <- apply(dtm_combined , 1, sum)
dtm_combined  <- dtm_combined[rowTotals> 0, ]
# Run LDA using Gibbs sampling
ldaOut_combined <-
  LDA(dtm_combined, k, method="Gibbs", control=list(nstart=nstart,
                                                    seed = seed,
                                                    best=best,
                                                    burnin = burnin,
                                                    alpha=alpha,
                                                    delta=delta,
                                                    iter=iter,
                                                    thin=thin))
  # theta_dj;
  # Probabilities associated with each topic assignment for each record
theta_combined <- as.data.frame(ldaOut_combined@gamma, rownames(dtm_combined))
```

```
  # phi_mj
  # the probability of using word m in topic j
phi_combined <- posterior(ldaOut_combined)$terms

save(theta_combined,file="../output/theta_combined.RData")
save(phi_combined,file="../output/phi_combined.RData")
```

Now we have the data ready for clustering (theta values). We view each document as a point in the k dimensional latent space and we can form a distance matrix between documents based on the euclidean distance metric

Now from here our approach deviates from the original paper. In the paper, the author proposed to create a similarity matrix (string distance between author names)

The use of the similarity matrix is to be able to differentiate different authors in the clustering process. So that documents with different authors names automatically get put into different clusters.

However, we elect not to use the similarity matrix because we can simply filter out author names instead. The reason they can't filter in the original paper is because the same author can have variations to their names in an uncleaned data set. But since our data set consists of the same representation for every name we don't need to do so. (ex. there is no A.J.Gupta)

The second way we differ from the original paper is in the height parameter we choose. The author proposed epsilon = 0.05, Epsilon limits how far clusters have to be to be considered different clusters. However due to the fact that we have much less words in each documents than they did in the paper (they had the first page of each document while we only had the tile and coauthors). We have documents that are much less similar in its latent space representation.

A good way of seeing the reason behind this is noting the following scenerio. Two papers about machine learning can have completely different titles. But they probably share words like "probability" "statistic" "regression" etc in the actual document. So through cross validation, we picked epsilon to be 0.15.

```
load("../output/theta_combined.RData")
load("../output/theta_list.RData")


#clustering on the combined theta list (from 1 dtm)
#create a column of names (14 distinct ones) to filter dtm by names later
theta_combined$name<-  sub("(.*?)_.*", "\\1", row.names(theta_combined))
name.list <- unique(theta_combined$name)
cluster.list.combined=NULL
for (i in name.list){
  print(i)
  Theta <- filter(theta_combined, theta_combined$name == i)
  #Theta<- as.matrix(as.data.frame(theta_combined[[3]][2]))
  Theta<- as.matrix(as.data.frame(Theta[,-ncol(Theta)]))
  distance<-dist(Theta, method = "euclid")
  cluster <- hclust(distance, method = "complete")
  #cut tree at h= 0.15, differ from the 0.05 proposed. read explanation in doc
  cut <- cutree(cluster, h=0.15)
  unique(cut)
  cluster.list.combined[[which(name.list== i)]] <- cut
}

save(cluster.list.combined,file="../output/cluster_list_combined.RData")
```

```
#clustering on the theta formed from the non-combined dtm
cluster.list.separate=NULL
for (i in 1:length(theta_list)){
  print(i)
  Theta<- as.matrix(as.data.frame(theta_list[[i]][2]))
  Theta<- Theta[,-11]
  distance<-dist(Theta, method = "euclid")
  cluster <- hclust(distance, method = "complete")
  cut <- cutree(cluster, h=0.15)
  unique(cut)
  cluster.list.separate[[i]] <- cut
}
save(cluster.list.separate,file="../output/cluster_list_separate.RData")
```

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)):

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.

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

The results that we managed to obtain was not ideal - although we did better than random, we had a low precision. The reasons why we think this is the case is because we had a very limited data set to use to differentiate/identify authors, i.e. we only had 20+ words for each document and we are projecting it onto a 10 dimensional space (selected via cross validation). We could have improved the results by taking the first page of each document being identified as elaborated by paper 4. Another possible reason is that we didn't take advantage of the structure of the data and lumped all the words together into 1 DTM.

```
source('../lib/evaluation_measures.R')
load("../output/cluster_list_separate.RData")
load("../output/cluster_list_combined.RData")
name_list<-unlist(strsplit(row.names(theta_combined),"_"))
name_ID <- data.frame(matrix(name_list, ncol=3, byrow=T)[,1:2])
names(name_ID)<-c("Name", "AuthorID")

performance.list.separate=list()
```

```
for (i in 1:length(theta_list)){
  G<-name_ID$AuthorID[name_ID$Name==query.list[i]]
  M<-cluster.list.separate[[i]]
  matching_matrix_separate <- matching_matrix(G,M)
  performance <- performance_statistics(matching_matrix_separate)
  performance.list.separate[[i]]<-performance
}

performance.separate <- data.frame(matrix(unlist(performance.list.separate), ncol=3, byrow=T))

save(performance.list.separate,file="../output/performance.list.separate.RData")
performance.separate<-data.frame(matrix(unlist(performance.list.separate),ncol=4,byrow=TRUE))
names(performance.separate)<-c("precision","recall","f1","accuracy" )
save(performance.separate,file="../output/performance_separate_df.RData")


performance.list.combined=list()
for (i in 1:length(theta_list)){
  G<-name_ID$AuthorID[name_ID$Name==query.list[i]]
  M<-cluster.list.combined[[i]]
  matching_matrix_combined <- matching_matrix(G,M)
  performance <- performance_statistics(matching_matrix_combined)
  performance.list.combined[[i]]<-performance
}

save(performance.list.combined,file="../output/performance.list.combined.RData")
performance.combined<-data.frame(matrix(unlist(performance.list.combined),ncol=4,byrow=TRUE))
names(performance.combined)<-c("precision","recall","f1","accuracy" )
save(performance.combined,file="../output/performance_combined_df.RData")


perf_s<-colMeans(performance.separate)
perf_c<-colMeans(performance.combined)
compare_df <- data.frame(perf_s,perf_c)
names(compare_df)<-c("Separate","Combined")
kable(compare_df,caption="Comparision of performance for two LDA methods",digits = 2)
```

Table 1: Comparision of performance for two LDA methods

	Separate	Combined
precision	0.26	0.29
recall	0.24	0.26
f1	0.22	0.25
accuracy	0.79	0.81