Project 4 - Group 6

Zeyu Gan, Virgile Mison, Galen Simmons, Siyuan Yao, Qingyuan Zhang

4/14/2017

Setup: assign Knitr root directory and load dependencies

We set the `knitr` root.dir to the project directory and load/install the necessary packages to run our `main.Rmd` script.

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: NLP

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## Loading required package: slam
```

Section I: Paper 3

"Name Disambiguation in Author Citations using a K-way Spectral Clustering Method"

As discussed in `Section 3.1`, authors Han, Zha, and Giles use **three** citation attributes to design features for name disambiguation. Those attributes are:

- co-author names
- paper titles
- publication venue titles

Together these attributes are called a "citation vector." For a dataset with m features, each citation is representated as an m-dimensional vector given by $M = (\alpha_1, \ldots, \alpha_m)$, where α_i is the weight assigned to feature i. Two types of feature weights assignments are profiled: (i) TFIDF and (ii) normalized TF ("NTF").

We demonstrate how we create the citation vector from our clean data sources in the `output` library in the chunks below:

Step 1: Load text file

```
source(file.path(projDir,"lib","feature_extraction.R"))
exFile <- file.path(projDir,"output","Agupta.csv")
exText <- readTextFile(exFile)
exText %>%
  tbl_df() %>%
  select(QuestAuthor, Coauthor, Paper, Journal) %>%
  head()
```

```
## # A tibble: 6 × 4
##   QuestAuthor                                                Coauthor
##         <chr>                                                   <chr>
## 1     A Gupta                                                    <NA>
## 2     A Gupta                                           A Acero; Y Rui
## 3     A Gupta                                       A Acharya; M Tambe
## 4     A Gupta                                                A Agarwal
## 5     A Gupta                                A Agrawal; N Chaddha; T Meng
## 6     A Gupta A Balachandran; E Sanocki; G Jancke; J Grudin; J Cadiz
## # ... with 2 more variables: Paper <chr>, Journal <chr>
```

Step 2: Prep the co-author and journal terms to be included in the document term matrix

We want the individual co-author names and journal names to be considered as single "terms" in our corpus. Therefore, we collapse the spaces separating the unique letters in someone's name, so that it appears to be a term. For example, "C L Zhang" would become "clzhang". Likewise, with journal names, we combine them into a single string without spaces so that each journal is a unique term in our Document Term Matrix.

```
as_tibble(exText) %>%
    mutate(x = str_replace_all(Coauthor, " ", "")) %>%
    mutate(x = str_replace_all(x, ";", " ")) %>%
    mutate(y = str_replace_all(Journal, " ", "")) %>%
    mutate(term_col = tolower(paste(x, y, Paper))) %>%
    select(term_col) %>%
    head()
```

```
## # A tibble: 6 × 1
##                                                                  term_col
##                                                                     <chr>
## 1 na parleparallelarchitecturesandlanguageseurope stanford dash multiprocesso
## 2 aacero yrui acmmultimedia automatically extracting highlights for tv baseba
## 3 aacharya mtambe ieeetransparalleldistribsyst implementation of production s
## 4 aagarwal sigmetricsmeasurementandmodelingofcomputersystems memory-reference
## 5 aagrawal nchaddha tmeng icmcsinternationalconferencemultimediacomputingands
## 6 abalachandran esanocki gjancke jgrudin jcadiz cscwconferencecomputersupport
```

Step 3: Run our spectral clustering method on the example dataset

We use the `tm` package to construct citation vectors in the manner described above. We then use the `matching_matrix` and `performance_statistics` functions in the `lib/evaluation_metrics.R` file to benchmark our results.

Details of Gaussian-similarity-kernel and k-means clustering implementation

First, we compute the simmilarity between citations from the TF-IDF or NTF matrix of citations. We use a Gaussian kernel as a measure of similarity. Then, we create an undirected graph based on the similarities to extract some manifold in the data, we thereby obtain A, the affinity matrix. After, we calculate the degree matrix D (diagonal) where each diagonal value is the degree of the respective vertex (*e.g.* sum of rows). We compute the unnormalized graph Laplacian:

$$U = D - A$$

Then, assuming that we want k clusters, we find the k smallest eigenvectors of U. This represents the points in a new k-dimensional space with low-variance. Finally, in this transformed space, it becomes easy for a standard k-means clustering to find the appropriate clusters.

Details of spectral clustering algorithm

From the TF-IDF or NTF matrix of citations, we compute the cosine similarity between each citation vectors as follows:

$$similarity = cos(\theta) = \frac{a \cdot b}{\|a\| \, \|b\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}}$$

This matrix is called the **Gram matrix** A. In the first step of the algorithm, we determine the k largest eigenvectors of A: X_k, a n-by-k matrix. Each row of X_k corresponds to a citation vector. Then, we compute the **QR decomposition with column pivoting** applied to ${X_k}^T$, *e.g.* we find the matrices P (permutation matrix, n-by-n), Q (orthogonal, k-by-k) and R (left-upper-triangular, k-by-n), so that:

$${X_k}^T P = QR = Q[R_{11}, R_{12}]$$

R_{11} will be the k-by-k upper-triangular matrix. We then compute the matrix $\hat{R}$:

$$\hat{R} = R_{11}^{-1} R P^T = R_{11}^{-1}[R_{11}, R_{12}]P^T = [I_k, R_{11}^{-1} R_{12}]P^T$$

Finally, the cluster membership of each citation vector is determined by the row index of the largest element in absolute value of the corresponding column of $\hat{R}$.

```
source(file.path(projDir,"lib","SpectralClustering.R"))
source(file.path(projDir,"lib","evaluation_measures.R"))

# start counter - estimated time 5 sec
start.time <- Sys.time()

num_authors <- length(unique(exText$AuthorID))
author_id <- exText$AuthorID
exDTM <- createCitationMatrix(exText)

exTFIDF <- weightTfIdf(exDTM, normalize = FALSE)
exNTF <- weightTf(exDTM)

exResultsTFIDF <- runSpectralClusteringQR(exTFIDF, num_authors)
exResultsNTF <- runSpectralClusteringQR(exNTF, num_authors)

m0 <- matching_matrix(author_id,exResultsTFIDF)
m1 <- matching_matrix(author_id,exResultsNTF)

p0 <- performance_statistics(m0)
p1 <- performance_statistics(m1)
```

```
# end counter
end.time <- Sys.time()
study.time <- end.time - start.time
kable(data.frame(study_time = study.time))
```

study_time

6.877977 secs

Step 4: View results from sample study

We examine some results from our sample methods to confirm our methodology

```
ex_df <- data.frame(
    study=rep("Agupta", 2),
    method=c("QR Spectral Clustering - TFIDF",
             "QR Spectral Clustering - NTF"),
    precision=c(p0$precision, p1$precision),
    recall=c(p0$recall, p1$recall),
    f1=c(p0$f1, p1$f1),
    accuracy=c(p0$accuracy, p1$accuracy),
    mcc=c(p0$mcc, p1$mcc)
    )

kable(ex_df)
```

study	method	precision	recall	f1	accuracy	mcc
Agupta	QR Spectral Clustering - TFIDF	0.1267663	0.2457352	0.1672528	0.7591650	0.0462060
Agupta	QR Spectral Clustering - NTF	0.1078698	0.2760624	0.1551253	0.7040427	0.0184078

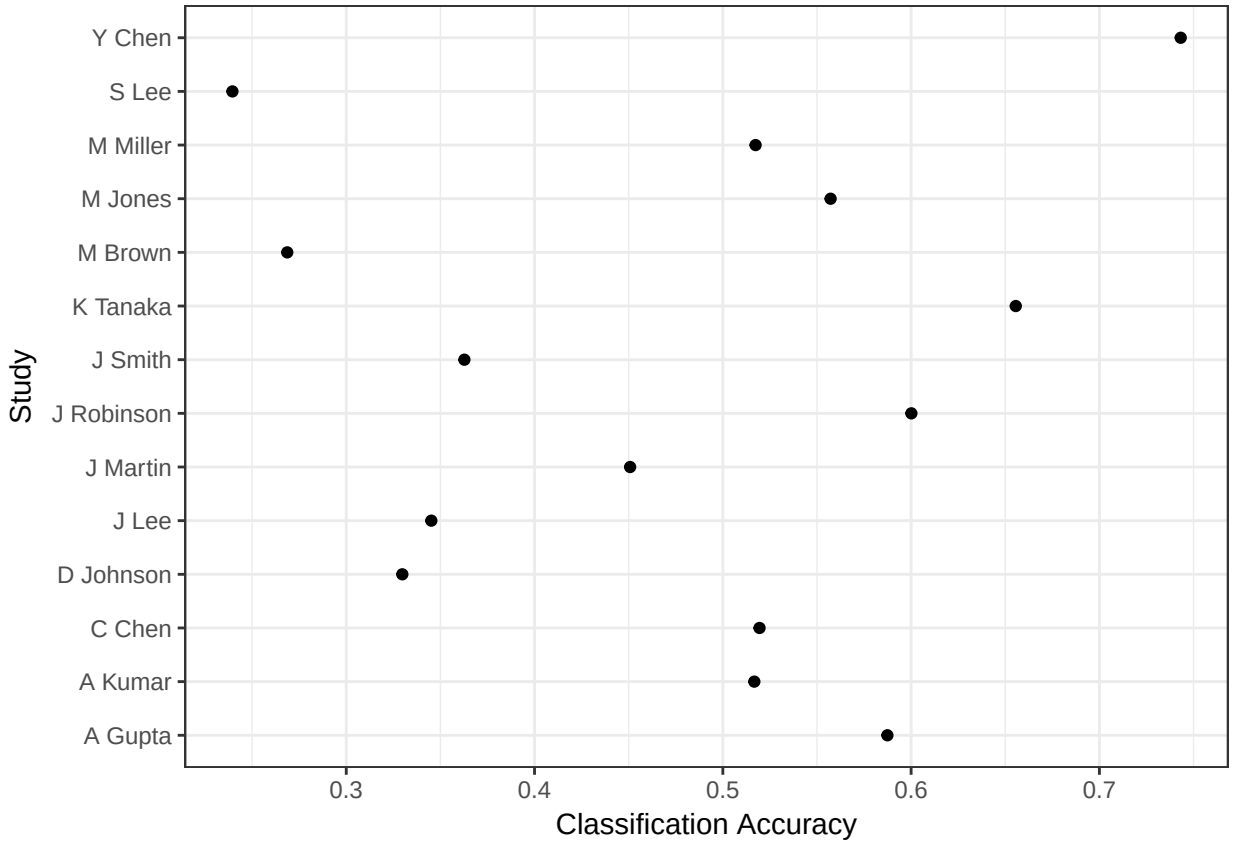
Step 5: Load results from all studies

```
load(file.path(projDir,"output/processed_data/prelimResults.RData"))
library(ggplot2)
```

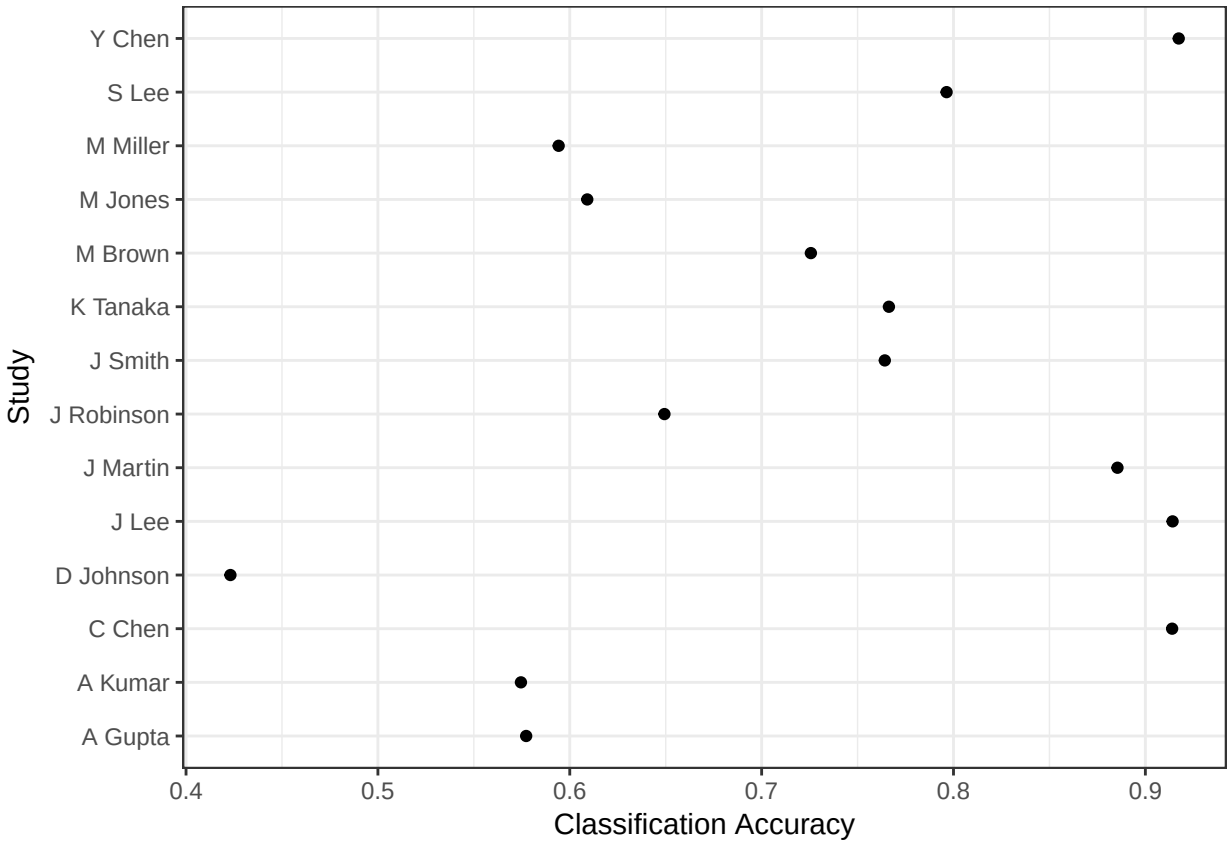
```
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:NLP':
##
##     annotate
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
tb %>%
  transform(accuracy  = as.numeric(accuracy)) %>%
  arrange(desc(accuracy)) %>%
  ggplot(aes(x=accuracy, y=study)) +
  geom_point() +
```

```
  theme_bw() +
  xlab('Classification Accuracy') +
  ylab('Study')
```




```
tb_teacher%>%
  transform(accuracy  = as.numeric(accuracy)) %>%
  arrange(desc(accuracy)) %>%
  ggplot(aes(x=accuracy, y=study)) +
  geom_point() +
  theme_bw() +
  xlab('Classification Accuracy') +
  ylab('Study')
```




```
# needs implementation
```

Supplemental notes on Paper #3

Term Frequency Reverse Document Frequency (TF-IDF) Why TF-IDF? Term frequency counts the number of times a word from a corpus apprears in a particular document. Some words appear frequently in all documents (i.e. they are common locally). Other words appear rarely in the corpus (i.e. they are rare globally). Inverse document frequency weights words that appear rarely more heavily by applying the following transformation to the term frequency (tf) vector:

$$log \cdot \frac{\#ofdocs}{1 + \#ofdocsusingword}$$

Distance metrics

Scaled Euclidean Distance:

$$d(x_i, x_q) = \sqrt{a_1(x_i[1] - x_q[1])^2 + \ldots + a_d(x_i[d] - x_q[d])^2}$$

Cosine Similarity:

$$\frac{x_i^T x_q}{\|x_i\| \, \|x_q\|} = cos(\theta)$$

Complexity of brute-force search

Given a query point, $O(N)$ distance computations per 1-NN query. By extension, $O(N \log k)$ distance computations per k-NN query.

K-means algorithm

1. Initialize cluster centers
2. Assign observations to closest cluster center
3. Revise cluster centers as mean of assigned observations
4. Repeat steps 1 & 2 until convergence

Failure modes of k-means

- disparate cluster sizes
- overlapping clusters
- different shaped / oriented clusters

Section II: Paper 6
"A Constraint-Based Probabilistic Framework for Name Disambiguation"
Step 1: Load and process the data
```
r AKumar <- read.csv(file.path(projDir,"output","AKumar.csv")) AGupta <-
read.csv(file.path(projDir,"output","Agupta.csv")) CChen <-
read.csv(file.path(projDir,"output","CChen.csv")) DJohnson <-
read.csv(file.path(projDir,"output","DJohnson.csv")) JLee <-
read.csv(file.path(projDir,"output","JLee.csv")) JMartin <-
read.csv(file.path(projDir,"output","JMartin.csv")) JRobinson <-
read.csv(file.path(projDir,"output","JRobinson.csv")) JSmith <-
read.csv(file.path(projDir,"output","JSmith.csv")) KTanaka <-
read.csv(file.path(projDir,"output","KTanaka.csv")) YChen <-
read.csv(file.path(projDir,"output","YChen.csv")) SLee <-
read.csv(file.path(projDir,"output","SLee.csv")) MMiller <-
read.csv(file.path(projDir,"output","MMiller.csv")) MJones <-
read.csv(file.path(projDir,"output","MJones.csv")) MBrown <-
read.csv(file.path(projDir,"output","MBrown.csv"))
```
"'r # Write a function to calcuate the DTM for paper dtm_paper <- function(df) { it_train <- itoken(as.character(df$Paper), preprocessor = tolower, tokenizer = word_tokenizer, ids = nrow(df), # turn off progressbar because it won't look nice in rmd progressbar = FALSE) vocab <- create_vocabulary(it_train, stopwords = c("a", "an", "the", "in","on","at", "of", "above", "under")) # Now that we have a vocabulary list, we can construct a document-term matrix. vectorizer <- vocab_vectorizer(vocab) dtm_train <- create_dtm(it_train, vectorizer) return(dtm_train) }
Write a function to calcuate the DTM for coauthor dtm_coauthor <- function(df) { it_train <- itoken(as.character(df$Coauthor), preprocessor = tolower, tokenizer = word_tokenizer, ids = nrow(df), # turn off progressbar because it won't look nice in rmd progressbar = FALSE) vocab <- create_vocabulary(it_train) # Now that we have a vocabulary list, we can construct a document-term matrix. vectorizer <- vocab_vectorizer(vocab) dtm_train <- create_dtm(it_train, vectorizer) return(dtm_train) }
Write a function to calcuate the DTM for Journal dtm_journal <- function(df) { it_train <- itoken(as.character(df$Journal), preprocessor = tolower, tokenizer = word_tokenizer, ids = nrow(df), # turn off progressbar because it won't look nice in rmd progressbar = FALSE) vocab <- create_vocabulary(it_train) # Now that we have a vocabulary list, we can construct a document-term matrix. vectorizer <- vocab_vectorizer(vocab) dtm_train <- create_dtm(it_train, vectorizer) return(dtm_train) } "'
Then, we want to use DTM to compute TF-IDF transformation on DTM.
"'r dtm_paper_tfidf <- function(df) { tfidf <- TfIdf$new() return(fit_transform(dtm_paper(df), tfidf)) }

dtm_coauthor_tfidf <- function(df) { tfidf <- TfIdf$new() return(fit_transform(dtm_coauthor(df), tfidf)) }
dtm_journal_tfidf <- function(df) { tfidf <- TfIdf$new() return(fit_transform(dtm_journal(df), tfidf)) }
# We want to combine the three citations together dtm_tfidf <- function(df) { total <- cbind(dtm_paper_tfidf(df), dtm_coauthor_tfidf(df), dtm_journal_tfidf(df)) return(total) }
# pap <- dtm_paper_tfidf(AGupta) # coa <- dtm_coauthor_tfidf(AGupta) # jor <- dtm_journal_tfidf(AGupta)
compute cosine similarities matrix (Gram Matrix) docsdissim <- function(df) { cosSparse(t(dtm_tfidf(df))) } "'
Step 3: Clustering
Following suggestion in the paper 6, we carry out hierarchical clustering method under the cosine distance as the baseline method. The number of clsters is assumed known as stated in the paper.
"'r basemodel <- function(df) { cossim <- docsdissim(df) rownames(cossim) <- c(1:nrow(dtm_tfidf(df))) colnames(cossim) <- c(1:nrow(dtm_tfidf(df)))
h <- hclust(as.dist(cossim), method = "ward.D") result_hclust <- cutree(h,length(unique(df$AuthorID))) return(result_hclust) }
basemodel_AGupta <- basemodel(AGupta) basemodel_AKumar <- basemodel(AKumar) # start.time <- Sys.time() # cossim <- docsdissim(AGupta) # rownames(cossim) <- c(1:nrow(dtm_tfidf(AGupta))) # colnames(cossim) <- c(1:nrow(dtm_tfidf(AGupta))) #compute pairwise cosine similarities using cosSparse function in package qlcMatrix # h <- hclust(as.dist(cossim), method = "ward.D") # result_hclust <- cutree(h,length(unique(AGupta$AuthorID))) # end.time <- Sys.time() # time_hclust <- end.time - start.time # table(result_hclust) "'
"'r matching_matrix_hclust_AGupta <- matching_matrix(AGupta$AuthorID,basemodel_AGupta) performance_hclust_AGupta <- performance_statistics(matching_matrix_hclust_AGupta) matching_matrix_hclust_AKumar <- matching_matrix(AKumar$AuthorID,basemodel_AKumar) performance_hclust_AKumar <- performance_statistics(matching_matrix_hclust_AKumar) compare_df <- data.frame(author=c("AGupta","AKumar"), precision=c(performance_hclust_AGupta$precision, performance_hclust_AKumar$precision), recall=c(performance_hclust_AGupta$recall, performance_hclust_AKumar$recall), f1=c(performance_hclust_AGupta$f1, performance_hclust_AKumar$f1), accuracy=c(performance_hclust_AGupta$accuracy, performance_hclust_AKumar$accuracy)) kable(compare_df,caption="Comparision of performance for two clustering methods",digits = 2) "'

Table: Comparision of performance for two clustering methods

author	precision	recall	f1	accuracy
AGupta	0.08	0.04	0.05	0.86
AKumar	0.19	0.07	0.10	0.74