Project 4 - Example Main Script

Jing Wu, Tian Zheng

3/22/2017

Step 0: Setup Knitr root directory

Make sure that this chunk outputs the project folder **not** the full path to the `doc` folder.

```
require("knitr")
```

```
## Loading required package: knitr
```

```
opts_knit$set(root.dir = normalizePath(".."))
projDir <- opts_knit$get("root.dir")
projDir
```

```
## [1] "/Users/galen/Google Drive/Applied Data Science Projects/Proj4/Spr2017-proj4-team6"
```

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")
```

```
## Loading required package: pacman
```

```
pacman::p_load(text2vec, dplyr, qlcMatrix, kernlab, knitr)

#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. You may need to find regular matched in the input string vectors by using regex in R. Here is a tutorial for regular expression in R, which might help you https://rstudio-pubs-static.s3.amazonaws.com/74603_76cd14d5983f47408fdf0b323550b846.html

```
AKumar <- data.frame(
      scan(
      file.path(projDir,"data","nameset","AKumar.txt", fsep = .Platform$file.sep),
      what = list(Coauthor = "", Paper = "", Journal = ""),
      sep=">",
      quiet=TRUE),
    stringsAsFactors=FALSE)
# This need to be modified for different name set
# extract canonical author id befor "_"
AKumar$AuthorID <- sub("_.*","",AKumar$Coauthor)
# extract paper number under same author between "_" and first whitespace
AKumar$PaperNO <- sub(".*_(\\w*)\\s.*", "\\1", AKumar$Coauthor)
# delete "<" in AKumar$Coauthor, you may need to further process the coauthor
# term depending on the method you are using
AKumar$Coauthor <- gsub("<","",sub("^.*?\\s","", AKumar$Coauthor))
# delete "<" in AKumar$Paper
AKumar$Paper <- gsub("<","",AKumar$Paper)
# add PaperID for furthur use, you may want to combine all the nameset files and
# then assign the unique ID for all the citations
AKumar$PaperID <- rownames(AKumar)
```

Step 2: Feature design

Following the section 3.1 in the paper, we want to use paper titles to design features for citations. As the notation used in the paper, we want to find a m-dimensional citation vector α_i for each citation i, $i = 1, ..., n$. In this dataset, $n = 244$. We study "TF-IDF" (term frequency-inverse document frequency) as suggested in the paper.

TF-IDF is a numerical statistics that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, text mining, and user modeling. The TF-IDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

$$\begin{aligned}
\text{TF}(t) &= \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}} \\
\text{IDF}(t) &= \log\frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}} \\
\text{TF-IDF}(t) &= \text{TF}(t) \times \text{IDF}(t)
\end{aligned}$$

To compute TF-IDF, we first need to construct a document-term matrix (DTM). In other words, the first step is to vectorize text by creating a map from words to a vector space. There are some good packages you could use for text mining (probably you have tried during first project, you don't need to follow my code if you are familiar with other package), e.g. *text2vec, tm, tidytext.* Here, we are going to use *text2vec* package. A good tutorial can be found here, https://cran.r-project.org/web/packages/text2vec/vignettes/text-vectorization.html.

Let's first create a vocabulary-based DTM. Here we collect unique terms from all documents and mark each of them with a unique ID using the `create_vocabulary()` function. We use an iterator to create the vocabulary.

```
it_train <- itoken(AKumar$Paper,
             preprocessor = tolower,
             tokenizer = word_tokenizer,
             ids = AKumar$PaperID,
             # 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"))

vocab
```

```
## Number of docs: 244
## 9 stopwords: a, an, the, in, on, at ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##              terms terms_counts doc_counts
##   1:       modular            1          1
##   2: interference            1          1
##   3:      assembly            1          1
##   4:     tolerance            1          1
##   5:          form            1          1
##  ---
## 662:         based           40         38
## 663:         yeast            1          1
## 664:       elastic            2          2
## 665:       schemes            1          1
## 666:   information            3          3
```

Here, we remove pre-defined stopwords, the words like “a”, “the”, “in”, “I”, “you”, “on”, etc, which do not provide much useful information.

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)
```

Now we have DTM and can check its dimensions.

```
dim(dtm_train)
```

```
## [1] 244 666
```

As you can see, the DTM has 244 rows, equal to the number of citations, and 666, equal to the number of unique terms excluding stopwords.

Then, we want to use DTM to compute TF-IDF transformation on DTM.

```
tfidf <- TfIdf$new()
dtm_train_tfidf <- fit_transform(dtm_train, tfidf)
```

Step 3: Clustering

Following suggestion in the paper, we carry out spectral clustering on the Gram matrix of the citation vectors by using R function `specc()` in *kernlab*. The number of clusters is assumed known as stated in the paper.

```
start.time <- Sys.time()
result_sclust <- specc(as.matrix(dtm_train_tfidf),
                       centers=length(unique(AKumar$AuthorID)))
end.time <- Sys.time()
time_sclust <- end.time - start.time
table(result_sclust)
```

```
## result_sclust
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## 12 12  7 31  9 19 14 17 54 21 10 14 16  8
```

We can also using hierarchical clustering method under the cosine distance. The intention of using a different clustering method is just to let you know how to compare performance between various methods.

```
start.time <- Sys.time()
docsdissim <- cosSparse(t(dtm_train_tfidf))
```

```r
rownames(docsdissim) <- c(1:nrow(dtm_train_tfidf))
colnames(docsdissim) <- c(1:nrow(dtm_train_tfidf))
#compute pairwise cosine similarities using cosSparse function in package qlcMatrix
h <- hclust(as.dist(docsdissim), method = "ward.D")
result_hclust <- cutree(h,length(unique(AKumar$AuthorID)))
end.time <- Sys.time()
time_hclust <- end.time - start.time
table(result_hclust)
```

```
## result_hclust
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## 44 29 39 22 18 16 16 12 13  7  6  8  5  9
```

Step 4: 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)):

Matching matrix for the agreement between two sets of clusters

		Gold standard clusters (*G*)	
		Match	Mismatch
Machine-generated clusters (*M*)	Match	a	b
	Mismatch	c	d

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.

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

```r
source('lib/evaluation_measures.R')
matching_matrix_hclust <- matching_matrix(AKumar$AuthorID,result_hclust)
performance_hclust <- performance_statistics(matching_matrix_hclust)
matching_matrix_sclust <- matching_matrix(AKumar$AuthorID,result_sclust)
performance_sclust <- performance_statistics(matching_matrix_sclust)
compare_df <- data.frame(method=c("sClust","hClust"),
                         precision=c(performance_sclust$precision, performance_hclust$precision),
                         recall=c(performance_sclust$recall, performance_hclust$recall),
                         f1=c(performance_sclust$f1, performance_hclust$f1),
                         accuracy=c(performance_sclust$accuracy, performance_hclust$accuracy),
                        time=c(time_sclust,time_hclust))
kable(compare_df,caption="Comparision of performance for two clustering methods",digits = 2)
```

Table 1: Comparision of performance for two clustering methods

method	precision	recall	f1	accuracy	time
sClust	0.32	0.15	0.21	0.75	1.07620502 secs
hClust	0.20	0.09	0.12	0.73	0.02515912 secs