Version 5/11/2010

**ReadMe Primer**

In this primer, we do three things.

* Run the example provided in the ReadMe package.
* Run a slightly different example using the same data source.
* Run ReadMe using data of our choosing. This requires a number of additional steps, and we can thank Loren for writing code that makes this possible!

**Read me requires Python**. You need to confirm that you have Python on your computer. In Windows, search for Python and confirm that you can access it from your command line. Otherwise, download Python.

This step involves downloading Python from <http://www.python.org/> . In Windows, **DO NOT DOWNLOAD THIS FILE TO** “C:\Program Files\Python”. The space is Program Files is likely to break the path. So download the Python to C:\Python. Or C:\Python2.6 depending on the version you download.

To run ReadMe you need to have Python on your variable path. This can be done two ways. The first is to use the command line. Type Meta-R (Windows button) and then in the Run window write cmd and click ok or press enter. The command line is open now and then type:

set PATH=%PATH%;c:\python

or the path where you installed Python. Then you need to restart your computer. Upon restart, check that Python is set on your path by opening up the command line and typing python. Python should now run. You will know because it will say Python 2.6.6 and some stuff about help and copyright. To exit out type exit() and you are out of python and can close the command line.

**Open R**

library")

library(ReadMe)

Might as well install these packages now! Don’t forget to load them in your library as well.

quadprog

xtable

gmodels

----------------------------------------------------------------------------------------

**Running Gary King's Example**

Create a Working Directory that matches where the ReadMe folder is located on your computer (your path may be different from what is specified here)

final <- "C:/R/R-2.10.1/library/ReadMe"

setwd(final)

Go to that working directory, and then to go to the demo working directory (this should be the correct path)

oldwd <- getwd()

setwd(system.file("demofiles/clintonposts",package="ReadMe"))

library(ReadMe)

**Now you are ready to run the ReadMe example. The example partitions a set of cases (blog posts) into train and test sets, and then predicts the labels of the test set.** *Running ReadMe, may take a little while, depending on size of the dataset.*

undergrad.results <- undergrad(sep=',', pyexe="C:/Python25")

*This may give a warning message about python not being on path, but should be ok.*

undergrad.preprocess <- preprocess(undergrad.results)

readme.results <- readme(undergrad.preprocess)

head(readme.results)

*You should see some results?*

**Now we create an output table that compares the ‘true’ label proportions of the blog test set (i.e. assigned by experts) with the ReadMe estimated label proportions. In addition, we generate a plot that shows that the errors are unbiased.**

true <-readme.results$true.CSMF

estimate <- readme.results$est.CSMF

comb <- data.frame(cbind(true,estimate))

comb$trumin <- comb$true - comb$estimate

Now you can look at it by typing

comb

----------------------------------------------------------------------------------------

**Another Example Run**

In this example, we re-run the results after first excluding less common words (i.e. those that are not found in at least 20 percent of the blogs.

setwd("C:/Users/Loren/R/win-library/2.9/ReadMe/demofiles")

output <- undergrad(threshold=.2, sep=',', pyexe="C:/Python25")

----------------------------------------------------------------------------------------

**Importing a different database into ReadMe**

First, find the ReadMe/demofiles/clintonposts folder in your directory. Open one of the numbered text files. Each of these files contains the text of a blog. Scroll to the bottom and open the control.txt file. The first column indicates the text file to be referenced, the second is the true label for that text file, and the third indicates whether that case is to be used for training (1) or testing (0).

This is a very different structure than we saw for TextTools (the train and test tables were in a single Access database). So using ReadMe may require converting data from one form to another. Loren Collingwood has written some code (in the TextTools R package) to do this.

*Assuming that you have already installed TextTools on your computer, load it.*

library(texttoolsb)

Then create an R data frame (‘data1’) by identifying a datasource and the relevant table within that datasource. In this case, I am grabbing the data from the “traintable” of the Access database “machinelearn.accdb” that I have placed in the listed directory. It could be located somewhere else. If you want to use the same database, it’s at faculty.washington.edu/jwilker/559.

data1 <- datagrab("machinelearn1.accdb","traintable",

path="C:/R/R-2.10.1/library/texttoolsb/TextTools\_V0.11")

Now we create a new object (data2) that is just the text found in column 4 of data1.

data2 <- as.matrix(data1[,4])

setwd("C:/R/R-2.10.1/library/ReadMe")

Now we make a new ReadMe directory and change into that

shell('mkdir readme\_bills')

setwd(paste(getwd(),"readme\_bills", sep="\\"))

Then we loop over the title column to place each row into its own List

titlesub <- as.list(rep(NA,nrow(data2)))

for (i in 1:nrow(data2)) {titlesub[[i]] <- data2[i,]}

Then we write each list object (bill title) to its own text file in the working directory just created

n <- length(data2) #n is the length of the dataframe, so here n=6287

for (i in 1:n) {

a <- data.frame(data2[[i]])

myfile <- gsub("()","",paste(i, ".txt",sep=""))

#include the sepcall here so you don’t get spaces. Write.table writes each row to its own .txt file.

write.table(a, file=myfile, sep="", row.names=F,col.names=F, quote=F, append=F)

}

And then we create the first column (IDs) of the Control.txt file as discussed above

myfile <- as.vector(c(rep(NA,n)))

for (i in 1:n) {

mytime <- format(Sys.time(),"%b\_%d\_%H\_%M\_%S\_%Y")

myfile[i] <- gsub("()","",paste(i, ".txt",sep=""))

}

Specify which cases are to be used for training by using the repeat command. In this case we create a vector of 0’s and 1’s where half the bills are training bills (1), half are test bills (0).

TRAININGSET <- c(rep(1,round(n/2)),rep(0,round(n/2)-1))

Add the labels (found in column 2 of the original Access table) to the control.txt file, by using the cbind command.

Myfile2 <- data.frame(myfile)

controlfile <- cbind(myfile,data1[2],TRAININGSET)

Name the columns of control.txt and then write out the control.txt file, which ReadMe needs in order to run its analysis.

colnames(controlfile) <- c("ROWID","TRUTH","TRAININGSET")

control2 <- as.matrix(controlfile)

write.table(control2,file="control.txt",

sep=",",row.names=FALSE,quote=FALSE)

*To confirm that this worked, access the directory you created and check that a numbered file contains a bill title and that the second column of Control.text file contains the labels. Be sure to close the files!!*

--------------------------------------------------------------------------------------

**Running ReadMe on the new data**

Now we are ready to switch from TextTools to ReadMe to run the program on this new dataset of bills. Running ReadMe, may take a little while, depending on size of Dataset.

library(ReadMe)

library(quadprog)

undergrad.results <- undergrad(sep=',', pyexe="C:/Python25")

*Or whatever your path is. May give a warning message about python not being on path, but should be ok.*

undergrad.preprocess <- preprocess(undergrad.results)

readme.results <- readme(undergrad.preprocess)

head(readme.results)

*You should see some results?*

**Again, create an output table that compares the ‘true’ label proportions of the blog test set (i.e. assigned by experts) with the ReadMe estimated label proportions.**

true <-readme.results$true.CSMF

estimate <- readme.results$est.CSMF

comb <- data.frame(cbind(true,estimate))

comb$trumin <- comb$true - comb$estimate

Now you can look at it by typing

comb

------------------------------------------------------------------------------------------------

**Comparing ReadMe proportion estimates to TextTools proportion estimates.**

This may go beyond your interests. The goal is to compare the Readme proportion estimates to the proportions generated by TextTools. Before the code below can be of use, we have to train and test the TextTools algorithms using the same datasource.

This code then draws on those results generate label proportions for each of the TextTools algorithms that can be compared to the ‘true’ label proportions from the same table. The TextTools library must be loaded

library(texttoolsb)

setwd(system.file("TextTools\_V0.11", package="texttoolsb"))

newdat <- datagrab("machinelearn.accdb", "testtable")

*note that this is a second Access database (not machinelearn.accb)*

library(gmodels) #you may need to install this?

svm1 <- CrossTable(newdat$new\_code\_svm)

svm2 <- t(svm1$prop.row)

*we are not going to do lingpipe in this example*

#ling1 <- CrossTable(newdat$new\_code\_ling)

#ling2 <- t(ling1$prop.row)

max1 <- CrossTable(newdat$new\_code\_maxent)

max2 <- t(max1$prop.row)

naive1 <- CrossTable(newdat$new\_code\_naive)

naive2 <- t(naive1$prop.row)

**Now we put them together so that we can compare the ReadMe and TextTools proportion results**

true <-readme.results$true.CSMF

estimate <- readme.results$est.CSMF

comb <- data.frame(cbind(true,estimate))

comb3 <- cbind(comb,svm2,max2,naive2)

colnames(comb3) <- c("True","ReadMe","SVM", "MaxEnt", "Naive")

**To look at the table**

comb3

*In the output, the first column is the “true” label proportion. The succeeding columns are the estimated proportions, beginning with ReadMe.*

----------------------------------------------------------------------------------------

**For writing tables to LaTex (otherwise copy and paste)**

library(xtable) (may have to install this package)

xtable(comb3) #if you don’t use LaTex then don’t need this.

#Order the table based on True-Estimate ReadMe scores.

comb4 <- comb3[order(comb3[,'True - Estimate']),]

**Plot to confirm that error estimate is unbiased.**

pdf(file="simulation.pdf")

plot(density(rnorm(10000,mean(comb3[3]),sd(comb3[3]))), main="Simulation of Prediction Error",xlab="Error Estimate", ylab="Density")

dev.off()

Unadulterated R code

###R Ready Code for the example above. Note that you will need to upload the two Access databases as described above.

#1. RUN THE README EXAMPLE

library(ReadMe)

library(quadprog)

library(gmodels)

final <- "C:/R/R-2.10.1/library/ReadMe"

oldwd <- getwd()

setwd(system.file("demofiles/clintonposts",package="ReadMe"))

library(ReadMe)

undergrad.results <- undergrad(sep=',', pyexe="C:/Python25")

undergrad.preprocess <- preprocess(undergrad.results)

readme.results <- readme(undergrad.preprocess)

head(readme.results)

true <-readme.results$true.CSMF

estimate <- readme.results$est.CSMF

comb <- data.frame(cbind(true,estimate))

comb$trumin <- comb$true - comb$estimate

comb

#2. RUN A SECOND README EXAMPLE

setwd("C:/Users/Loren/R/win-library/2.9/ReadMe/demofiles")

output <- undergrad(threshold=.2, sep=',', pyexe="C:/Python25")

#3. CONVERT DATA IN AN ACCESS TABLE (MACHINELEARN1) TO THE FILE STRUCTURE NEEDED TO RUN README

library(texttoolsb)

data1 <- datagrab("machinelearn1.accdb","traintable",

path="C:/R/R-2.10.1/library/texttoolsb/TextTools\_V0.11")

data2 <- as.matrix(data1[,4])

setwd("C:/R/R-2.10.1/library/ReadMe")

shell('mkdir readme\_bills')

setwd(paste(getwd(),"readme\_bills", sep="\\"))

titlesub <- as.list(rep(NA,nrow(data2)))

for (i in 1:nrow(data2)) {titlesub[[i]] <- data2[i,]}

n <- length(data2) #n is the length of the dataframe, so here n=6287

for (i in 1:n) {

a <- data.frame(data2[[i]])

myfile <- gsub("()","",paste(i, ".txt",sep=""))

write.table(a, file=myfile, sep="", row.names=F,col.names=F, quote=F, append=F)

}

myfile <- as.vector(c(rep(NA,n)))

for (i in 1:n) {

mytime <- format(Sys.time(),"%b\_%d\_%H\_%M\_%S\_%Y")

myfile[i] <- gsub("()","",paste(i, ".txt",sep=""))

}

TRAININGSET <- c(rep(1,round(n/2)),rep(0,round(n/2)-1))

Myfile2 <- data.frame(myfile)

controlfile <- cbind(myfile,data1[2],TRAININGSET)

colnames(controlfile) <- c("ROWID","TRUTH","TRAININGSET")

control2 <- as.matrix(controlfile)

write.table(control2,file="control.txt",

sep=",",row.names=FALSE,quote=FALSE)

#4. RUN README ON THIS NEW DATASET

library(ReadMe)

library(quadprog)

undergrad.results <- undergrad(sep=',', pyexe="C:/Python25")

undergrad.preprocess <- preprocess(undergrad.results)

readme.results <- readme(undergrad.preprocess)

head(readme.results)

true <-readme.results$true.CSMF

estimate <- readme.results$est.CSMF

comb <- data.frame(cbind(true,estimate))

comb$trumin <- comb$true - comb$estimate

comb

#5. COMPARE README RESULTS TO TEXTTOOLS RESULTS

library(texttoolsb)

setwd(system.file("TextTools\_V0.11", package="texttoolsb"))

newdat <- datagrab("machinelearn.accdb", "testtable")

library(gmodels) #you may need to install gmodels?

svm1 <- CrossTable(newdat$new\_code\_svm)

svm2 <- t(svm1$prop.row)

#ling1 <- CrossTable(newdat$new\_code\_ling) #Not going to do LingPipe

#ling2 <- t(ling1$prop.row)

max1 <- CrossTable(newdat$new\_code\_maxent)

max2 <- t(max1$prop.row)

naive1 <- CrossTable(newdat$new\_code\_naive)

naive2 <- t(naive1$prop.row)

true <-readme.results$true.CSMF

estimate <- readme.results$est.CSMF

comb <- data.frame(cbind(true,estimate))

comb3 <- cbind(comb,svm2,max2,naive2)

colnames(comb3) <- c("True","ReadMe", "SVM", "MaxEnt", "Naive")

comb3

pdf(file="simulation.pdf")

plot(density(rnorm(10000,mean(comb3[3]),sd(comb3[3]))), main="Simulation of Prediction Error",xlab="Error Estimate", ylab="Density")

dev.off()