#!/usr/bin/Rscript

# The following R code is designed to clean and format the data in

# preparation for throwing it into Weka.

# Files (assumed to be tab-delimited) are read into R data frames.

# Instances, corresponding to individual tokens, are ordinally labeled

# according to the sentence they belong to. Blank and marker instances are

# discarded. The class variable (Entity.ID) is cleaned of extraneous characters

# and transformed from ~400 numerical types to 7 character types, as specified

# by the problem. Lemma Reference Uncertainty, Speaker’s Most Frequent Referent,

# Previous Speaker, Next Speaker, Previous Lemma, Second Previous Lemma,

# Next Lemma, Second Next Lemma, and Implied Gender features are generated.

# Unneeded features are discarded, and all String and Character features are transformed

# into Factor features. Finally, the data frame is written out in both .csv and .arff format,

# the latter being the preferred format for Weka classification.

# Enable libraries

install.packages("data.table",repos = "<http://cran.us.r-project.org>“)

install.packages("stringr",repos = "<http://cran.us.r-project.org>")

install.packages("foreign",repos = "<http://cran.us.r-project.org>")

install.packages("arules",repos = "<http://cran.us.r-project.org>")

install.packages("plyr",repos = "<http://cran.us.r-project.org>“)

library("data.table")

library("stringr")

library("foreign")

library("arules")

library("plyr")

# README Step 2

# Reads in txt file as a tab-delimited "csv-as-far-as-R-is-concerned" file with headers

# (Assumption: the txt file has been converted to tab-delimited format)

data <- read.csv("file.txt", header=TRUE, sep="\t")

# README Step 3

# Label each sentence in the file with a sentence number so that blank rows can be

# removed without loss of information

j = 1;

for (i in 1:nrow(data)){

if (data$SeasonEpisode[i] == ""){

j = j + 1;

}

data$Sentence[i] = j;

}

# Discard blank rows and rows containing only begin/end of document markers

data <- subset(data, SeasonEpisode != "")

data <- subset(data, SeasonEpisode != "#end document")

data <- subset(data, SeasonEpisode != "#begin document")

# README Step 4

# Clean Entity.ID of extraneous characters

data$Entity.ID <- str\_extract(data$Entity.ID, "[[:digit:]]+")

# Transform Entity.ID such that if the referenced character is not one of

# the 6 main characters, they are categorized as "Other". Otherwise, the

# categorization is transformed into the character's name.

data$Entity.ID[!is.na(data$Entity.ID) & data$Entity.ID != 183 &

data$Entity.ID != 306 & data$Entity.ID != 292 &

data$Entity.ID != 335 & data$Entity.ID != 248 &

data$Entity.ID != 59] <- "Other"

data$Entity.ID[data$Entity.ID == 183] <- "Joey Tribbiani"

data$Entity.ID[data$Entity.ID == 306] <- "Rachel Green"

data$Entity.ID[data$Entity.ID == 292] <- "Phoebe Buffay"

data$Entity.ID[data$Entity.ID == 335] <- "Ross Geller"

data$Entity.ID[data$Entity.ID == 248] <- "Monica Geller"

data$Entity.ID[data$Entity.ID == 59] <- "Chandler Bing"

# Convert "data frame" to a "data table" for business logic reasons

data.dt<-as.data.table(data)

# README Step 5

# Clean the name of the Speaker (sometimes includes extraneous characters and spaces,

# remove some structural instances)

data.dt$Speaker <- str\_extract(data.dt$Speaker, "[a-zA-Z0-9\_.]+")

data.dt <- subset(data.dt, Speaker != "All")

data.dt <- subset(data.dt, Speaker != "NAME")

# README Step 6

# Feature: Lemma Reference Uncertainty

# First, gather a list of all lemma types

lemmalist <- unique(data$Lemma)

lemma.df <- data.frame(lemmalist)

# Tabulate the number of unique referents that lemma refers to (most lemmas reference nobody and will

# have scores of zero, while lemmas like "you" will refer to a variety of characters)

for (i in 1:length(lemmalist)){

lemma.df$uniquereferents[i] <- length(unique(data$Entity.ID[data$Lemma == lemma.df$lemmalist[i] & !is.na(data$Entity.ID)]))

}

# Bin the number of unique referents into 3 ordinal categories

lemma.df$Lemma\_Reference\_Uncertainty <- discretize(lemma.df$uniquereferents, categories = 3, labels = c("1","2","3"))

lemma.dt<-as.data.table(lemma.df)

lemma.dt$Lemma<-lemma.dt$lemmalist

# Join the new Lemma Reference Uncertainty field to the dataset

data.dt <- join(data.dt, lemma.dt[, .(Lemma, Lemma\_Reference\_Uncertainty)], by = "Lemma")

# README Step 7

# Feature: Speaker's Most Frequent Referent

# First, gather a list of all speaker types

Speaker<-unique(data.dt$Speaker)

speakermfr.dt<-as.data.table(Speaker)

# Then, for each unique speaker, create a table of all characters they refer to, sort it, and take the top entry

# as their most frequent referent

for (i in 1:length(Speaker)){

speakermfr.dt$Speaker\_MFR[i] <- names(sort(table(data.dt$Entity.ID[data.dt$Speaker == speakermfr.dt$Speaker[i]]), decreasing = TRUE)[1])

}

# Join the new field to the dataset

data.dt <- join(data.dt, speakermfr.dt[, .(Speaker, Speaker\_MFR)], by = "Speaker")

# README Step 8

# Feature: Previous Speaker

# Loop forward through the sequential instances in the dataset, storing the "previous speaker" as a variable

# that begins empty. At each iteration, check whether the speaker has changed, and update "previous speaker"

# and "current speaker" variables if so. Then, write the previous speaker name into the "previous speaker" field.

prev<-"NA"

cur<-data.dt$Speaker[1]

for (i in 1:nrow(data.dt)){

if (data.dt$Speaker[i] == cur){

data.dt$Prev\_Speaker[i]<-prev

}

else{

prev<-cur;

cur<-data.dt$Speaker[i]

data.dt$Prev\_Speaker[i]<-prev

}

}

# Feature: Next Speaker

# As Previous Speaker, but in reverse. Loop backwards through the sequential instances in the dataset,

# and whenever the speaker name changes, update "next speaker" and "current speaker" variables, then write

# the next speaker's name into the new "next speaker" field.

nextspkr<-"NA"

cur<-data.dt$Speaker[nrow(data.dt)]

for (i in 1:nrow(data.dt)){

j <- nrow(data.dt) + 1 - i;

if (data.dt$Speaker[j] == cur){

data.dt$Next\_Speaker[j]<-nextspkr

}

else{

nextspkr<-cur;

cur<-data.dt$Speaker[j]

data.dt$Next\_Speaker[j]<-nextspkr

}

}

# README Step 9

# Feature: Previous and Second Previous Lemma

# As above, with speakers, only without the necessity of checking whether the value of the field has

# changed. Store previous two lemmas in variables, update with each forward-loop iteration, and write

# the previous two lemmas into two new fields.

prev<-"NA"

prev2<-"NA"

for (i in 1:nrow(data.dt)){

cur<-as.character(data.dt$Lemma[i])

data.dt$Prev\_Lemma[i]<-prev;

data.dt$Second\_Prev\_Lemma[i]<-prev2

prev2<-prev;

prev<-cur;

}

# Feature: Next and Second Next Lemma

# Identical to Previous Lemma procedure, but in reverse, looping backward and writing the next two lemmas

# into two new fields.

nextlem<-"NA"

nextlem2<-"NA"

for (i in 1:nrow(data.dt)){

j<- nrow(data.dt) + 1 - i;

cur<-as.character(data.dt$Lemma[j])

data.dt$Next\_Lemma[j]<-nextlem;

data.dt$Second\_Next\_Lemma[j]<-nextlem2;

nextlem2<-nextlem;

nextlem<-cur;

}

# README Step 10

# Implied Gender

# Primitive measure of the "implied gender" of pronoun reference lemmas. If the lemma is "she" or "her",

# note the implied gender as "F", and if the lemma is "he" or "him", note the implied gender as "M". Otherwise,

# populate the field with an NA.

for (i in 1:nrow(data.dt)){

if (data.dt$Lemma[i] == "she" | data.dt$Lemma[i] == "her"){

data.dt$Implied\_Gender[i]<-"F";

}

else if (data.dt$Lemma[i] == "he" | data.dt$Lemma[i] == "him"){

data.dt$Implied\_Gender[i]<-"M";

}

else{

data.dt$Implied\_Gender[i]<-"NA";

}

}

# README Step 11

# Remove fields that won't be used in analysis

data.dt$Frameset.ID<-NULL

data.dt$Word.Sense<-NULL

data.dt$Named.Entity.Tag<-NULL

data.dt$Constituency.Tag<-NULL

# README Step 12

# Convert all character/string features to factors so that, when written to .arff format, they

# will be interpreted as Nominal features rather than Strings.

data.dt$Speaker<-as.factor(data.dt$Speaker)

data.dt$Entity.ID<-as.factor(data.dt$Entity.ID)

data.dt$Speaker\_MFR<-as.factor(data.dt$Speaker\_MFR)

data.dt$Prev\_Speaker<-as.factor(data.dt$Prev\_Speaker)

data.dt$Next\_Speaker<-as.factor(data.dt$Next\_Speaker)

data.dt$Prev\_Lemma<-as.factor(data.dt$Prev\_Lemma)

data.dt$Second\_Prev\_Lemma<-as.factor(data.dt$Second\_Prev\_Lemma)

data.dt$Next\_Lemma<-as.factor(data.dt$Next\_Lemma)

data.dt$Second\_Next\_Lemma<-as.factor(data.dt$Second\_Next\_Lemma)

data.dt$Implied\_Gender<-as.factor(data.dt$Implied\_Gender)

#README Step 13

# Write to both a csv and an arff (the latter format for Weka)

write.csv(data.dt, file="file.csv", row.names=FALSE)

write.arff(data.dt, file="file.arff")