#######################################################

# Detecting temporal cognition in text: Comparison of judgements by self, expert and machine

# E.Walsh and J. Busby Grant

# Code: erin.walsh@anu.edu.au

# R version 3.2.0 (2015-04-16)

# Platform: x86\_64-w64-mingw32/x64 (64-bit)

# Running under: Windows 7 x64 (build 7601) Service Pack 1

#######################################################

######################## Libraries and functions

# Work

setwd("REMOVED FOR PRIVACY")

# Home

#setwd("REMOVED FOR PRIVACY")

library(psych) # version 1.5.8 for easy descriptives

# coreNLP::downloadCoreNLP() # One-time setup

library(coreNLP) # version 3.3.3Only works at home due to java dependency problems.

library(ggplot2)

library(gridExtra)

# Version 1: naive tense extraction.

snl\_tense\_extractor\_v1<-function(input\_text){future<-FALSE;present<-FALSE;past<-FALSE;other<-FALSE

if(grepl("/MD",input\_text)){future<-TRUE}

if(grepl("ing/NN",input\_text) |

grepl("/VBG",input\_text) |

grepl("/VBP",input\_text) |

grepl("/VBZ",input\_text) |

grepl("/UH",input\_text)

){present<-TRUE}

if(grepl("/VBD",input\_text) | grepl("/VBN",input\_text)){past<-TRUE}

if(!future & !present &!past & !input\_text==""){other<-TRUE}

out<-data.frame(text=input\_text,future, present,past,other)

return(out)}

# Version 2: naive tense extraction + explicit use of the term 'remember' or 'future' or any stems

snl\_tense\_extractor\_v2<-function(input\_text){future<-FALSE;present<-FALSE;past<-FALSE;other<-FALSE

if(grepl("/MD",input\_text)){future<-TRUE}

if(grepl("ing/NN",input\_text) |

grepl("/VBG",input\_text) |

grepl("/VBP",input\_text) |

grepl("/VBZ",input\_text) |

grepl("/UH",input\_text)

){present<-TRUE}

if(grepl("/VBD",input\_text) | grepl("/VBN",input\_text)){past<-TRUE}

# Addition: any actual reference to 'remembering'

if(grepl("remember",input\_text)){past<-TRUE}

# Addition: any actual reference to the future

if(grepl("future",input\_text)){future<-TRUE}

if(!future & !present &!past & !input\_text==""){other<-TRUE}

out<-data.frame(text=input\_text,future, present,past,other)

return(out)}

# Version 3: naive + ties split based on position in sentence, first word wins - allows an 'other' category

snl\_tense\_extractor\_v3<-function(input\_text){

input\_text<-as.character(input\_text)

if(is.na(input\_text)){return(NA)}

if(nchar(input\_text)==1){return(NA)}

splittit<-paste0("/",unlist(strsplit(input\_text,"/")))

futurepos<-c(grep("/MD",splittit),grep("future",splittit))

presentpos<-c(grep("ing/NN",splittit), grep("/NN",splittit),

grep("/VBG",splittit),

grep("/VBP",splittit),

grep("/VBZ",splittit),

grep("/UH",splittit))

pastpos<-c(grep("/VBD",splittit),

grep("/VBN",splittit),

grep("remember",splittit))

# Place back in order so we can pull out the first one

theorder<-data.frame( rbind(

cbind(rep("future", length(futurepos)),futurepos),

cbind(rep("present", length(presentpos)),presentpos),

cbind(rep("past", length(pastpos)),pastpos)))

out<-as.character(theorder[which.min(as.numeric(levels(theorder[,2]))[theorder[,2]]) ,1])

# Make the output dataframe; boolean test or if empty due to missingness coerce to false

future<-out=="future"; if(length(future)==0){future<-FALSE}

past<-out=="past"; if(length(past)==0){past<-FALSE}

present<-out=="present"; if(length(present)==0){present<-FALSE}

other<-(!future & !past & !present)

# Return

outdat<-data.frame(text=input\_text, future, present, past, other)

return(outdat)}

#######################################################

######################## Data and checks

# Self-report results

sr\_dat<-read.csv("data/self\_report.csv")

# Check it loaded correctly

head(sr\_dat)

length(unique(as.character(sr\_dat$moreinfo))) # 2570 unique entries

# Ensure the 'moreinfo' column is correctly recognised as character, not factor.

sr\_dat$moreinfo<-as.character(sr\_dat$moreinfo)

# Get rid of empty rows

sr\_dat<-sr\_dat[-which(sr\_dat$moreinfo==""),]

# Check

head(sr\_dat); nrow(sr\_dat) # 2885

# specify rater

sr\_dat$rater<-"Self"

# Remove empty rows

sr\_dat<-sr\_dat[-which(as.character(sr\_dat$moreinfo)==" "),] # 2884

# Collapse redundant categories

sr\_dat[which(!is.na(sr\_dat$remembering)),"past"]<-TRUE

sr\_dat[which(!is.na(sr\_dat$knowing)),"past"]<-TRUE

sr\_dat[which(!is.na(sr\_dat$present)),"present"]<-TRUE

sr\_dat[which(!is.na(sr\_dat$future)),"future"]<-TRUE

sr\_dat[which(!is.na(sr\_dat$imagining)),"imagining"]<-TRUE

sr\_dat<-sr\_dat[,c("moreinfo","past","present","future","other")]

# Researcher coded results

researcher\_noties<-read.csv("data/orientation\_coding\_EW.csv")

# Check it loaded correctly

head(researcher\_noties)

length(unique(as.character(researcher\_noties$moreinfo))) # 2569 unique entries

# Ensure the 'moreinfo' column is correctly recognised as character, not factor.

researcher\_noties$moreinfo<-as.character(researcher\_noties$moreinfo)

# Check

head(researcher\_noties);nrow(researcher\_noties) # 2885

researcher\_ties<-read.csv("data/orientation\_coding\_JBG.csv")

# Check it loaded correctly

head(researcher\_ties)

length(unique(as.character(researcher\_ties$moreinfo))) # 2569 unique entries

# Ensure the 'moreinfo' column is correctly recognised as character, not factor.

researcher\_ties$moreinfo<-as.character(researcher\_ties$moreinfo)

# Get rid of empty rows

head(researcher\_ties);nrow(researcher\_ties) # 2885

# Convert 'NA' to 'FALSE' and 1 to 'TRUE'

researcher\_noties[is.na(researcher\_noties)] <- FALSE

researcher\_noties[,c("Past","Present","Future","Other")]<-researcher\_noties[,c("Past","Present","Future","Other")]==1

researcher\_ties[is.na(researcher\_ties)] <- FALSE

researcher\_ties[,c("Past","Present","Future","Other")]<-researcher\_ties[,c("Past","Present","Future","Other")]==1

# Allow ties for 'researcher\_ties'

researcher\_ties$mixed<- rowSums(researcher\_ties[,c("Past", "Present", "Future", "Other")])>1

# All rownames to lowercase for later merging

names(researcher\_noties)<-tolower(names(researcher\_noties))

names(researcher\_ties)<-tolower(names(researcher\_ties))

# Specify rater

researcher\_noties$rater<-"Researcher, no ties allowed"

researcher\_ties$rater<-"Researcher,ties allowed"

# Remove empty rows

researcher\_noties<-researcher\_noties[-which(as.character(researcher\_noties$moreinfo)==" "),] # 2884

researcher\_ties<-researcher\_ties[-which(as.character(researcher\_ties$moreinfo)==" "),] # 2884

#######################################################

######################## Parsing 001: Stanford Natural Language Parser

# (URL for manual comparison is http://nlp.stanford.edu:8080/parser/index.jsp)

# Clean to remove untokenizable non-askii characters

to\_tokenize<-sr\_dat$moreinfo

Encoding(to\_tokenize) <- "latin1"

to\_tokenize\_ascii\_cleaned<-iconv(to\_tokenize, "latin1", "ASCII", sub="")

# Actually tokenize

# # Last run 27/3/2018

# # Initialize

initCoreNLP()

snl\_annotated<-lapply(to\_tokenize\_ascii\_cleaned,annotateString)

snl\_tokens<-lapply(snl\_annotated,getToken)

snl\_tokens\_string<-lapply(snl\_tokens,function(x){paste0(x$token,"/",x$POS)})

snl\_tokens\_string\_flat<-as.character(unlist(lapply(snl\_tokens\_string, function(x){paste0(x,collapse=" ")})))

# # Check

head(cbind(sr\_dat$moreinfo,snl\_tokens\_string\_flat))

tail(cbind(sr\_dat$moreinfo,snl\_tokens\_string\_flat))

# # Save

# save(snl\_annotated,snl\_tokens,snl\_tokens\_string,snl\_tokens\_string\_flat, file="data/snl\_tokens.RData")

# Load

load("data/snl\_tokens.RData")

# Run

snl\_tense\_extracted\_v1<-lapply(snl\_tokens\_string\_flat,snl\_tense\_extractor\_v1)

snl\_tense\_extracted\_v2<-lapply(snl\_tokens\_string\_flat,snl\_tense\_extractor\_v2)

snl\_tense\_extracted\_v3<-lapply(snl\_tokens\_string\_flat,snl\_tense\_extractor\_v3)

# Rbind and clean

snl\_v1<-do.call(rbind, snl\_tense\_extracted\_v1)

snl\_v1$text<-as.character(snl\_v1$text)

snl\_v1$rater<-"SNL, naive"

snl\_v2<-do.call(rbind, snl\_tense\_extracted\_v2)

snl\_v2$text<-as.character(snl\_v2$text)

snl\_v2$rater<-"SNL, anchor terms"

snl\_v3<-do.call(rbind, snl\_tense\_extracted\_v3)

snl\_v3$text<-as.character(snl\_v3$text)

snl\_v3$rater<-"SNL, anchor terms, ties broken by position (first wins)"

######################## SUTime temporal tagger

# As in Thorstad, R., & Wolff, P. (2018). A big data analysis of the relationship between future thinking and decision-making.

# Proceedings of the National Academy of Sciences, 115(8), E1740-E1748.

# Can't run it propely in R, so will use my own code + the web portal at http://nlp.stanford.edu:8080/sutime/process

# Named Entity Recognition, TimeGuidelines for Temporal Expression Annotation for English for TempEval 2010

# Formal citation: Angel X. Chang and Christopher D. Manning. 2012. SUTIME: A Library for Recognizing and Normalizing Time Expressions.

# 8th International Conference on Language Resources and Evaluation (LREC 2012).

snl\_ner\_string<-lapply(snl\_tokens,function(x){paste0(x$token,"/",x$NER)})

snl\_ner\_string\_temps<-grepl("/DATE",snl\_ner\_string)

just\_temporals<-to\_tokenize\_ascii\_cleaned[which(snl\_ner\_string\_temps)]

# Ran this through http://nlp.stanford.edu:8080/sutime/process

just\_temporals\_tagged<-read.csv("data/su\_time.csv")

just\_temporals\_tagged\_unique<-just\_temporals\_tagged[!duplicated(just\_temporals\_tagged$token), ]

# Pull out timex3 categories

timex3<-just\_temporals\_tagged\_unique$category

# Strip off explicit dates

timex3\_stripped<-gsub('[^a-zA-Z]', '', timex3)

timex3\_stripped

# timex3, pull out actual tags

tense<-vector()

for(i in 1:nrow(just\_temporals\_tagged\_unique)){

temp<-strsplit(as.character(just\_temporals\_tagged\_unique$timex3[i]),">")[[1]][2]

tense[i]<-strsplit(as.character(temp),"<")[[1]][1]}

snl\_ner\_out<-data.frame(moreinfo=sr\_dat$moreinfo,

temporal\_tags\_tense=NA,

temporal\_tags\_timex3=NA,

temporal\_tags\_timex3\_raw=NA)

just\_temporals\_tagged\_unique$timex3<-as.character(just\_temporals\_tagged\_unique$timex3)

# To keep track if multiple indicators are in same place

indices<-rep(NA,nrow(snl\_ner\_out))

# Stitch our indicators back on

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

snl\_ner\_out[which(grepl(tense[[i]],snl\_ner\_out$moreinfo)),"temporal\_tags\_tense"]<-tense[[i]]

snl\_ner\_out[which(grepl(tense[[i]],snl\_ner\_out$moreinfo)),"temporal\_tags\_timex3"]<-timex3\_stripped[[i]]

snl\_ner\_out[which(grepl(tense[[i]],snl\_ner\_out$moreinfo)),"temporal\_tags\_timex3\_raw"]<-just\_temporals\_tagged\_unique$timex3[i]

indices[which(grepl(tense[[i]],snl\_ner\_out$moreinfo))]<-indices[which(grepl(tense[[i]],snl\_ner\_out$moreinfo))]+1

}

# Tidy

snl\_ner\_out$multiples<-indices>1

snl\_ner\_out$rater<-"snl\_SUTIME"

# Where SU time tags were created, swap them in here.

snl\_ner\_out$moreinfo<-as.character(snl\_ner\_out$moreinfo)

snl\_ner\_out[which(!snl\_ner\_out$temporal\_tags\_timex3\_raw==""),"moreinfo"]<-

snl\_ner\_out[which(!snl\_ner\_out$temporal\_tags\_timex3\_raw==""),"temporal\_tags\_timex3\_raw"]

# Does not neatly give past/future, but rather relative days. Replace with our goal format.

table(tolower(snl\_ner\_out$temporal\_tags\_tense))

snl\_ner\_out$temporal\_tags\_cleaned<-NA

snl\_ner\_out$temporal\_tags\_cleaned[which(tolower(snl\_ner\_out$temporal\_tags\_tense) %in% c("about 5 minutes ago","past"))]<-"past"

snl\_ner\_out$temporal\_tags\_cleaned[which(tolower(snl\_ner\_out$temporal\_tags\_tense) %in% c("presently","now","present"))]<-"present"

snl\_ner\_out$temporal\_tags\_cleaned[which(tolower(snl\_ner\_out$temporal\_tags\_tense) %in% c("future","next week","the future","tomorrow"))]<-"future"

snl\_ner\_out$temporal\_tags\_cleaned[which(!tolower(snl\_ner\_out$temporal\_tags\_tense) %in%

c("about 5 minutes ago","past",

"presently","now","present",

"future","next week","the future","tomorrow"))]<-"other"

table(tolower(snl\_ner\_out$temporal\_tags\_cleaned))

#######################################################

#################################### BASIC COMPARISONS (28/03/2018)

compare\_single\_response<-function(x){

sr\_outcome<-names(sr\_dat)[which(sr\_dat[x,]==1)]

# If there is no outcome, it should be NA.

if(length(sr\_outcome)==0){sr\_outcome<-NA}

# If there is more than one self-reported orientation, make this missing.

if(length(sr\_outcome)==2){sr\_outcome<-NA}

sr\_to\_bind<-cbind(as.character(sr\_dat[x,"moreinfo"]),"Self", sr\_outcome)

noties\_to\_bind<-cbind(researcher\_noties[x,"moreinfo"], researcher\_noties[x,]$rater,

names(researcher\_noties)[which( researcher\_noties[x,]==1)])

ties\_to\_bind<-cbind(researcher\_ties[x,"moreinfo"], researcher\_ties[x,]$rater,

names(researcher\_ties)[which(researcher\_ties[x,]==1)])

if(length(noties\_to\_bind)==2){noties\_to\_bind<-cbind(noties\_to\_bind,NA)} # To fix some missingness glitches

if(length(ties\_to\_bind)==2){ties\_to\_bind<-cbind(ties\_to\_bind,NA)} # To fix some missingness glitches

snl\_v1\_to\_bind<-cbind(snl\_v1[x,"text"], snl\_v1[x,]$rater,names(snl\_v1)[which(snl\_v1[x,]==1)])

snl\_v2\_to\_bind<-cbind(snl\_v2[x,"text"], snl\_v2[x,]$rater,names(snl\_v2)[which(snl\_v2[x,]==1)])

snl\_v3\_to\_bind<-cbind(snl\_v3[x,"text"], snl\_v3[x,]$rater,names(snl\_v3)[which(snl\_v3[x,]==1)])

snl\_sutime\_to\_bind<-cbind(snl\_ner\_out[x,"moreinfo"], snl\_ner\_out[x,]$rater,snl\_ner\_out[x,]$temporal\_tags\_cleaned)

out<-rbind(sr\_to\_bind,noties\_to\_bind,ties\_to\_bind,snl\_v1\_to\_bind,snl\_v2\_to\_bind,snl\_v3\_to\_bind,snl\_sutime\_to\_bind)

rownames(out)<-NULL # To avoid annoying error

out<-data.frame(out)

names(out)<-c("Text/coded","Rater","Orientation")

# Re-order for easier reading

out<-out[,c("Orientation","Rater","Text/coded")]

# See which of the methods match the orientation provided by self-report

out$correct<-out[which(out$Rater=="Self"),"Orientation"] == out[,"Orientation"]

return(out)

}

single\_accrual<-list() # Set up accumulator

# There were 2884 unique responses.

for(i in 1:2884){single\_accrual[[i]]<-compare\_single\_response(i)}

# Remove those with missing self-report, as this is the benchmark.

SR\_is\_missing<- is.na(unlist(lapply(single\_accrual,"[", 1, 1)))

table(SR\_is\_missing)

# SR\_is\_missing

# FALSE TRUE

# 2505 379

single\_accrual\_selected<-single\_accrual[which(!SR\_is\_missing)]

length(single\_accrual\_selected)

# Collapse into seperate dataframes for descriptives and further analysis

code\_1\_self\_rated<-do.call(rbind,lapply(single\_accrual\_selected,function(x){x[which(x$Rater=="Self"),]}))

table(code\_1\_self\_rated$Orientation);head(code\_1\_self\_rated)

# Researcher

code\_2\_Researcher\_no\_ties\_allowed<-do.call(rbind,lapply(single\_accrual\_selected,function(x){x[which(x$Rater=="Researcher, no ties allowed"),]}))

table(code\_2\_Researcher\_no\_ties\_allowed$Orientation);head(code\_2\_Researcher\_no\_ties\_allowed)

code\_3\_Researcher\_ties\_allowed<-do.call(rbind,lapply(single\_accrual\_selected,function(x){x[which(x$Rater=="Researcher,ties allowed"),]}))

table(code\_3\_Researcher\_ties\_allowed$Orientation);head(code\_3\_Researcher\_ties\_allowed)

# SNL

code\_4\_SNL\_naive<-do.call(rbind,lapply(single\_accrual\_selected,function(x){x[which(x$Rater=="SNL, naive"),]}))

table(code\_4\_SNL\_naive$Orientation); head(code\_4\_SNL\_naive)

snl\_tense\_extractor\_v1("In/IN 2019/CD ,/, I/PRP will/MD have/VB rememberd/VBN this/DT example/NN.")

code\_5\_SNL\_anchor\_terms\_ties<-do.call(rbind,lapply(single\_accrual\_selected,function(x){x[which(x$Rater=="SNL, anchor terms"),]}))

table(code\_5\_SNL\_anchor\_terms\_ties$Orientation); head(code\_5\_SNL\_anchor\_terms\_ties)

snl\_tense\_extractor\_v2("In/IN 2019/CD ,/, I/PRP will/MD have/VB rememberd/VBN this/DT example/NN.")

code\_6\_SNL\_anchor\_no\_ties<-do.call(rbind,lapply(single\_accrual\_selected,function(x){x[which(x$Rater=="SNL, anchor terms, ties broken by position (first wins)"),]}))

table(code\_6\_SNL\_anchor\_no\_ties$Orientation); head(code\_6\_SNL\_anchor\_no\_ties)

snl\_tense\_extractor\_v3("In/IN 2019/CD ,/, I/PRP will/MD have/VB rememberd/VBN this/DT example/NN.")

code\_7\_SNL\_sutime<-do.call(rbind,lapply(single\_accrual\_selected,function(x){x[which(x$Rater=="snl\_SUTIME"),]}))

table(code\_7\_SNL\_sutime$Orientation); head(code\_7\_SNL\_sutime)

####################################################### Plots!

grid.arrange(ncol=1, heights=c(0.3,0.7),

ggplot(na.omit(code\_1\_self\_rated), aes(x=Orientation)) + geom\_bar() +

theme(legend.position = "none") + ggtitle("1) Self-report")

,

arrangeGrob(

ggplot(na.omit(code\_2\_Researcher\_no\_ties\_allowed), aes(x=Orientation, fill=correct)) + geom\_bar() +

theme(legend.position = "none") + ggtitle("2) Researcher A (no ties)") +

geom\_bar(data=code\_1\_self\_rated, aes(x=Orientation),color="black", fill=NA) +ylim(0,2000)

,

ggplot(na.omit(code\_3\_Researcher\_ties\_allowed[which(!code\_3\_Researcher\_ties\_allowed$Orientation=="mixed"),]), #Omit this for tidy

aes(x=Orientation, fill=correct)) + geom\_bar() +

theme(legend.position = "none") + ggtitle("3) Researcher B (ties allowed)") +

geom\_bar(data=code\_1\_self\_rated, aes(x=Orientation),color="black", fill=NA)+ylim(0,2000)

,

ggplot(na.omit(code\_4\_SNL\_naive), aes(x=Orientation, fill=correct)) + geom\_bar() +

theme(legend.position = "none") + ggtitle("4) SNL, naïve (ties allowed)") +

geom\_bar(data=code\_1\_self\_rated, aes(x=Orientation),color="black", fill=NA)+ylim(0,2000)

,

ggplot(na.omit(code\_5\_SNL\_anchor\_terms\_ties), aes(x=Orientation, fill=correct)) + geom\_bar() +

theme(legend.position = "none") + ggtitle("5) SNL, anchor terms (ties allowed)") +

geom\_bar(data=code\_1\_self\_rated, aes(x=Orientation),color="black", fill=NA)+ylim(0,2000)

,

ggplot(na.omit(code\_6\_SNL\_anchor\_no\_ties), aes(x=Orientation, fill=correct)) + geom\_bar() +

theme(legend.position = "none") + ggtitle("6) SNL, anchor terms (no ties allowed)") +

geom\_bar(data=code\_1\_self\_rated, aes(x=Orientation),color="black", fill=NA)+ylim(0,2000)

,

ggplot(na.omit(code\_7\_SNL\_sutime), aes(x=Orientation, fill=correct)) + geom\_bar() +

theme(legend.position = "none") + ggtitle("7) SNL, suTime") +

geom\_bar(data=code\_1\_self\_rated, aes(x=Orientation),color="black", fill=NA)

))

####################################################### Percentages!

round(prop.table(table(code\_1\_self\_rated$Orientation))\*100,2)

round(prop.table(table(code\_2\_Researcher\_no\_ties\_allowed$correct))\*100,2)

round(prop.table(table(code\_3\_Researcher\_ties\_allowed$correct))\*100,2)

round(prop.table(table(code\_4\_SNL\_naive$correct))\*100,2)

round(prop.table(table(code\_5\_SNL\_anchor\_terms\_ties$correct))\*100,2)

round(prop.table(table(code\_6\_SNL\_anchor\_no\_ties$correct))\*100,2)

round(prop.table(table(code\_7\_SNL\_sutime$correct))\*100,2)

# Overall percentages.

round(prop.table(table(c(code\_2\_Researcher\_no\_ties\_allowed$correct,

code\_3\_Researcher\_ties\_allowed$correct))\*100),2)

round(prop.table(table(c(code\_4\_SNL\_naive$correct,

code\_5\_SNL\_anchor\_terms\_ties$correct,

code\_6\_SNL\_anchor\_no\_ties$correct,

code\_7\_SNL\_sutime$correct))\*100),2)

round(prop.table(table(c(code\_2\_Researcher\_no\_ties\_allowed$correct,

code\_6\_SNL\_anchor\_no\_ties$correct))\*100),2)

# Ties not allowed

round(prop.table(table(code\_2\_Researcher\_no\_ties\_allowed$correct))\*100,2)

round(prop.table(table(code\_6\_SNL\_anchor\_no\_ties$correct))\*100,2)

# Ties allowed

round(prop.table(table(code\_3\_Researcher\_ties\_allowed$correct))\*100,2)

round(prop.table(table(code\_3\_Researcher\_ties\_allowed$correct,

code\_3\_Researcher\_ties\_allowed$Orientation))\*100,1)

round(prop.table(table(code\_4\_SNL\_naive$correct))\*100,2)

round(prop.table(table(code\_5\_SNL\_anchor\_terms\_ties$correct))\*100,2)