03b CSI online aphasia: Typing - Automatic answer classification

Kirsten Stark

3/30/2021

Load packages

```
rm(list = ls())

library(tidyr)
library(dplyr)
```

```
##
## 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
```

```
library(stringr)
library(stringdist)
```

```
##
## Attaching package: 'stringdist'

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
options( "encoding" = "UTF-8" )
```

Load data

```
# input
input = "pretest_long.csv"
```

```r
# input synonym/alternative naming list
alternatives = "naming_alternatives.csv"

# load data
df <- read.csv(here::here("data", "transient_data_files", input))

# load alternatives
alternatives <- read.csv(here::here("data", "supplementary_info", alternatives),
                         sep = ";")
```

Load functions from Stark (2021); https://github.com/kirstenstark/stringmatch_typed_naming

```r
#https://github.com/kirstenstark/stringmatch_typed_naming
source(here::here("scripts","code","stringmatch_typed_naming.R"))
```

Preprocess data, applying functions

1) Clean word ending: Delete last characters of typed words if those are space or enter

This function checks whether the last character(s) of the word entries is a space or is the typed word "Enter", and if so, deletes the last/the last five characters. If the last character(s) are neither of both, the word remains unchanged. The function can be used within dplyr's mutate function. Additionally, the function has the option to delete an alternative ending, while keeping " " and "Enter" at the end of the word.

As entries, the delete_ending function takes the column with the word entries and, optionally, a custom ending.

We can repeat applying this function if we want to keep deleting if Enter or space is repeated several times at the end of the word. The while loops stops as soon as none of the words has a space or Enter (or custom ending) at the end.

```r
isnotequal <- 1
df$word.c = currentupdate = df$word
while (isnotequal > 0) {
  df <- df %>% mutate(word.c = delete_ending(df$word.c))
  isnotequal <- sum(currentupdate != df$word.c, na.rm = TRUE)
  currentupdate <- df$word.c
}
```

2) Replace special keys (e.g. backspace, shift, etc.) by other characters (e.g. numbers)

Special characters such as Enter and Backspace are written as entire words. We want to replace these with identifiable numbers.

Function takes as entries the word entries, the keys to be changed, and the characters they should be replaced with.

With new data we may have to check whether participants used any other special keys.

```
oldnames <- c("Enter", "CapsLock", "Shift", "ArrowLeft", "ArrowRight", "Backspace", "Control")
newnames <- c("1", "2", "3", "4", "5", "6", "7")
df$word.c <- replace_special_chars(input = df$word.c, oldnames = oldnames, newnames = newnames)
```

```
## [1] "The pattern Enter has been replaced by the pattern 1."
## [1] "The pattern CapsLock has been replaced by the pattern 2."
## [1] "The pattern Shift has been replaced by the pattern 3."
## [1] "The pattern ArrowLeft has been replaced by the pattern 4."
## [1] "The pattern ArrowRight has been replaced by the pattern 5."
## [1] "The pattern Backspace has been replaced by the pattern 6."
## [1] "The pattern Control has been replaced by the pattern 7."
```

3) Compute finally submitted words by applying all backspaces

Function takes as input the word entries and, optionally, the backspace identifier.

```
df$word.c <- replace_backspace(df$word.c, backspace = "6")
```

4) Compute fuzzy string matching (string distance) between word entries and items/alternatives by relying on the stringdist()-package

Calculate stringdistance between (backspace corrected) input word and item/alternative namings, and select the "best match", i.e. the item/alternative with the lowest distance and the first letter being correct.
The default method is the Jaro distance (Jaro-Winkler distance ("jw") with p = 0). Other methods, of the stringdist function (van der Loo, 2014) are possible as well, but further options of the stringdist function might be necessary to adapt as well.

Compute Jaro distance

```
#stringdist(toupper(df$word.c2[1:200]), toupper(df$item[1:200]), method = "jw")
tictoc::tic()
output <- calculate_stringdist(word = df$word.c, stims = df$item,
                               alternatives = alternatives,
                               method = "jw", p = 0,
                               firstlettercorrect = TRUE)
tictoc::toc()
```

```
## 0.186 sec elapsed
```

```
df$jaro <- output[,1]
df$bestmatch_jaro <- output[,2]
#df$jaro[1:200]
```

Alternatively: Compute Levenshtein distance (with all transformations being equally weighted)

```
#stringdist(toupper(df$word.c2[1:200]), toupper(df$item[1:200]), method = "lv")
tictoc::tic()
output <- calculate_stringdist(word = df$word.c, stims = df$item,
                               alternatives = alternatives,
                               method = "lv",
```

```r
                                weight = c(d = 1, i = 1, s = 1, t = 1),
                                firstlettercorrect = TRUE)
tictoc::toc()
```

```
## 0.08 sec elapsed
```

```r
df$lv <- output[,1]
df$bestmatch_lv <- output[,2]
#df$lv[1:200]
```

5) Classify word entries

Function that classifies the word entries for correctness and different typing errors.

Based on Jaro distance (d = 0.3)
The d-value indicates how many errors can be made (see van der Loo, 2014) and may be adapted for other study populations.

```r
df2 <- df %>%
  mutate(answer_auto_jaro = case_character_type(word, item,
          word.c, jaro, bestmatch_jaro, d = 0.3))
```

```r
df2 <- df2 %>%
  mutate(correct_auto_jaro = case_when(
    answer_auto_jaro == "correct" ~ 1,
    answer_auto_jaro == "correctedtocorrect" ~ 1,
    answer_auto_jaro == "approx_correct" ~ 1,
    answer_auto_jaro == "alternative" ~ 1,
    answer_auto_jaro == "alternative_corrected" ~ 1,
    answer_auto_jaro == "approx_alternative" ~ 1,
    TRUE ~ 0))
```

Based on Levenshtein distance (d = 1)
The d-value indicates how many errors can be made (see van der Loo, 2014) and may be adapted for other study populations.

```r
df2 <- df2 %>%
  mutate(answer_auto_lv = case_character_type(word, item,
          word.c, lv, bestmatch_lv, d = 3))
```

```r
df2 <- df2 %>%
  mutate(correct_auto_lv = case_when(
    answer_auto_lv == "correct" ~ 1,
    answer_auto_lv == "correctedtocorrect" ~ 1,
    answer_auto_lv == "approx_correct" ~ 1,
    answer_auto_lv == "alternative" ~ 1,
    answer_auto_lv == "alternative_corrected" ~ 1,
    answer_auto_lv == "approx_alternative" ~ 1,
    TRUE ~ 0))
```

Inspect results

Using the Jaro distance

```r
table(df2$answer_auto_jaro)
```

```
##
## alternative_corrected    approx_alternative        approx_correct
##                     5                     1                    18
##               correct    correctedtocorrect  distance_based_error
##                   116                     4                     1
##    first_letter_error                  isna           not_correct
##                     8                     5                     2
```

```r
table(df2$correct_auto_jaro)
```

```
##
##   0   1
##  16 144
```

```r
(incorrect_jaro <-df2 %>% filter(correct_auto_jaro == 0) %>%
    select(item, word, word.c, bestmatch_jaro, answer_auto_jaro))
```

```
##            item                                          word        word.c
## 1       kirsche               aBackspacekircBackspacescheEnter       kirsche
## 2        ameise                           sBackspaceameiseEnter        ameise
## 3         sense                                       actEnter           act
## 4        tempel                etBackspaceBackspacetempelEnter        tempel
## 5         tiger                                           <NA>          <NA>
## 6       leopard                                           <NA>          <NA>
## 7         gabel                                 hBackspacegabel         gabel
## 8       ohrring                                            ring          ring
## 9         feile                                            säge          säge
## 10 bohrmaschine                                   skkuschrauber  skkuschrauber
## 11  schornstein                                           <NA>          <NA>
## 12         fuss                                  gBackspacefuss          fuss
## 13        pferd                                            ferd          ferd
## 14       gondel kanuBackspaceBackspaceBackspaceBackspacegondle        gondle
## 15         baby                                           <NA>          <NA>
## 16  segelschiff                                           <NA>          <NA>
##    bestmatch_jaro     answer_auto_jaro
## 1         kirsche   first_letter_error
## 2          ameise   first_letter_error
## 3          SICHEL          not_correct
## 4          tempel   first_letter_error
## 5           tiger                 isna
## 6         leopard                 isna
## 7           gabel   first_letter_error
## 8     OHRANHÄNGER          not_correct
## 9        SPACHTEL distance_based_error
## 10  AKKUSCHRAUBER   first_letter_error
## 11    schornstein                 isna
```

```
## 12          fuss   first_letter_error
## 13         pferd   first_letter_error
## 14        gondel   first_letter_error
## 15          baby                 isna
## 16    segelschiff                isna
```

```
correct_jaro <-df2 %>% filter(correct_auto_jaro == 1) %>%
    select(item, word, word.c, bestmatch_jaro, answer_auto_jaro)
```

Using the Levenshtein distance

```
table(df2$answer_auto_lv)
```

```
##
## alternative_corrected    approx_alternative        approx_correct
##                     5                     1                    18
##               correct    correctedtocorrect    first_letter_error
##                   116                     4                     8
##                  isna           not_correct
##                     5                     3
```

```
table(df2$correct_auto_lv)
```

```
##
##   0   1
##  16 144
```

```
(incorrect_lv <-df2 %>% filter(correct_auto_lv == 0) %>%
    select(item, word, word.c, bestmatch_lv, answer_auto_lv))
```

```
##           item                                          word        word.c
## 1      kirsche               aBackspacekircBackspacescheEnter       kirsche
## 2       ameise                          sBackspaceameiseEnter        ameise
## 3        sense                                       actEnter           act
## 4       tempel               etBackspaceBackspacetempelEnter        tempel
## 5        tiger                                           <NA>          <NA>
## 6      leopard                                           <NA>          <NA>
## 7        gabel                                hBackspacegabel         gabel
## 8      ohrring                                           ring          ring
## 9        feile                                           säge          säge
## 10 bohrmaschine                                  skkuschrauber skkuschrauber
## 11  schornstein                                          <NA>          <NA>
## 12        fuss                                 gBackspacefuss          fuss
## 13       pferd                                           ferd          ferd
## 14      gondel kanuBackspaceBackspaceBackspaceBackspacegondle        gondle
## 15        baby                                           <NA>          <NA>
## 16  segelschiff                                          <NA>          <NA>
##     bestmatch_lv     answer_auto_lv
## 1        kirsche first_letter_error
## 2         ameise first_letter_error
## 3          sense        not_correct
```

```
## 4        tempel first_letter_error
## 5         tiger               isna
## 6       leopard               isna
## 7         gabel first_letter_error
## 8       ohrring        not_correct
## 9         feile        not_correct
## 10 AKKUSCHRAUBER first_letter_error
## 11   schornstein               isna
## 12          fuss first_letter_error
## 13         pferd first_letter_error
## 14        gondel first_letter_error
## 15          baby               isna
## 16   segelschiff               isna
```

```r
correct_lv <-df2 %>% filter(correct_auto_lv == 1) %>%
    select(item, word, word.c, bestmatch_lv, answer_auto_lv)
```

Comparing Jaro and Levensthein distance

```r
(differences <- df2 %>%
  filter((correct_auto_jaro==1&correct_auto_lv==0)|
           (correct_auto_jaro==1&correct_auto_lv==0)) %>%
  select(item, word, word.c, bestmatch_jaro, answer_auto_jaro,
         bestmatch_lv, answer_auto_lv,
         correct_auto_jaro, correct_auto_lv))
```

```
## [1] item             word             word.c           bestmatch_jaro
## [5] answer_auto_jaro bestmatch_lv     answer_auto_lv   correct_auto_jaro
## [9] correct_auto_lv
## <0 rows> (or 0-length row.names)
```

```r
nrow(differences)
```

```
## [1] 0
```

Typing error analyses based on automatic classification - JARO (d=0.3)

Amount of trials classified as correct and incorrect

```r
print("totaltrials:")
```

```
## [1] "totaltrials:"
```

```r
nrow(df2)
```

```
## [1] 160
```

```r
print("correct:")
```

```
## [1] "correct:"
```

```r
(correct = sum(df2$correct_auto_jaro == 1))
```

```
## [1] 144
```

```r
print("incorrect:")
```

```
## [1] "incorrect:"
```

```r
(incorrect = sum(df2$correct_auto_jaro == 0))
```

```
## [1] 16
```

Percentage of incorrect trials

```r
# incorrect/nrow(df2)*100
# incorrect/30/160*100
incorrect_per_subject <-
  as.data.frame(table(df2$subject, df2$correct_auto_jaro)) %>%
  filter(Var2 == 0) %>% select(Var1, Freq) %>%
  dplyr::rename(subject = Var1, perct_incorrect = Freq) %>%
  mutate(perct_incorrect = perct_incorrect/160)

print("Mean:")
```

```
## [1] "Mean:"
```

```r
round(mean(incorrect_per_subject$perct_incorrect)*100,2)
```

```
## [1] 10
```

```r
print("SD:")
```

```
## [1] "SD:"
```

```r
round(sd(incorrect_per_subject$perct_incorrect)*100,2)
```

```
## [1] NA
```

```r
print("Range:")
```

```
## [1] "Range:"
```

```r
round(range(incorrect_per_subject$perct_incorrect)*100,2)
```

```
## [1] 10 10
```

Correct/incorrect trials per participant:

```r
print(as.data.frame(table(
  df2$subject, df2$correct_auto_jaro == 1)) %>%
    filter(Var2 == TRUE) %>%
    dplyr::rename(subject = Var1, totaltrials = Var2,
                  correct_auto = Freq) %>%
    mutate(totaltrials = 160) %>%
    mutate(percentagecorrect = correct_auto/totaltrials))
```

```
##   subject totaltrials correct_auto percentagecorrect
## 1       1         160          144               0.9
```

Typing error analyses based on automatic classification - LEVENSHTEIN (d=3)

Amount of trials classified as correct and incorrect

```r
print("totaltrials:")
```

```
## [1] "totaltrials:"
```

```r
nrow(df2)
```

```
## [1] 160
```

```r
print("correct:")
```

```
## [1] "correct:"
```

```r
(correct = sum(df2$correct_auto_lv == 1))
```

```
## [1] 144
```

```r
print("incorrect:")
```

```
## [1] "incorrect:"
```

```r
(incorrect = sum(df2$correct_auto_lv == 0))
```

```
## [1] 16
```

Percentage of incorrect trials

```r
# incorrect/nrow(df2)*100
# incorrect/30/160*100
incorrect_per_subject <-
  as.data.frame(table(df2$subject, df2$correct_auto_lv)) %>%
  filter(Var2 == 0) %>% select(Var1, Freq) %>%
  dplyr::rename(subject = Var1, perct_incorrect = Freq) %>%
  mutate(perct_incorrect = perct_incorrect/160)

print("Mean:")
```

```
## [1] "Mean:"
```

```
round(mean(incorrect_per_subject$perct_incorrect)*100,2)
```

```
## [1] 10
```

```
print("SD:")
```

```
## [1] "SD:"
```

```
round(sd(incorrect_per_subject$perct_incorrect)*100,2)
```

```
## [1] NA
```

```
print("Range:")
```

```
## [1] "Range:"
```

```
round(range(incorrect_per_subject$perct_incorrect)*100,2)
```

```
## [1] 10 10
```

Correct/incorrect trials per participant:

```
print(as.data.frame(table(
  df2$subject, df2$correct_auto_lv == 1)) %>%
    filter(Var2 == TRUE) %>%
    dplyr::rename(subject = Var1, totaltrials = Var2,
                  correct_auto = Freq) %>%
    mutate(totaltrials = 160) %>%
    mutate(percentagecorrect = correct_auto/totaltrials))
```

```
##   subject totaltrials correct_auto percentagecorrect
## 1       1         160          144               0.9
```

Write data file for statistical analyses

```
write.csv(df2, here::here("data","transient_data_files", "data_long_final.csv"))
```