Package 'SemNetCleaner'

September 27, 2019

Title An Automated Cleaning Tool for Semantic and Linguistic Data

Version 1.1.1

Date 2019-09-27

Maintainer Alexander P. Christensen <alexpaulchristensen@gmail.com>

Description Implements several functions that automates the cleaning and spell-checking of text data. Also converges, finalizes, removes plurals and continuous strings, and puts text data in binary format for semantic network analysis. Uses the 'SemNetDictionaries' package to make the cleaning process more accurate, efficient, and reproducible.

Depends R (>= 3.5.0), SemNetDictionaries (>= 0.1.3)

License GPL (>= 3.0)

URL https://github.com/AlexChristensen/SemNetCleaner

BugReports https://github.com/AlexChristensen/SemNetCleaner/issues

NeedsCompilation no

Encoding UTF-8

LazyData true

Imports hunspell, searcher, stringdist, tcltk, foreign, readxl, R.matlab

RoxygenNote 6.1.1

Author Alexander P. Christensen [aut, cre] (<https://orcid.org/0000-0002-9798-7037>)

Repository CRAN

Date/Publication 2019-09-27 18:50:02 UTC

R topics documented:

`SemNetCleaner-package` *SemNetCleaner–package*

Description

Implements several functions that automates the cleaning and spell-checking of text data. Also converges, finalizes, removes plurals and continuous strings, and puts text data in binary format for semantic network analysis. Uses the `SemNetDictionaries` package to make the cleaning process more accurate, efficient, and reproducible.

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

See Also

Useful links:

- `https://github.com/AlexChristensen/SemNetCleaner`
- Report bugs at `https://github.com/AlexChristensen/SemNetCleaner/issues`

bad.response *Bad Responses to NA*

Description

A wrapper function to determine whether responses are good or bad. Bad responses are replaced with missing (NA). Good responses are returned.

Usage

```
bad.response(word, ...)
```

Arguments

word	Character. A word to be tested for whether it is bad
...	Vector. Additional responses to be considered bad

Value

If response is bad, then returns NA. If response is valid, then returns the response

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Bad response
bad.response(word = " ")

# Good response
bad.response(word = "hello")

# Make a good response bad
bad.response(word = "hello","hello")

# Add additional bad responses
bad.response(word = "hello", c("hello","world"))
```

`best.guess` *Makes Best Guess for Spelling Correction*

Description

A wrapper function for the best guess of a spelling mistake based on the letters, the ordering of those letters, and the potential for letters to be interchanged. The Damerau-Levenshtein distance is used to guide inferences into what word the participant was trying to spell from a dictionary (see `SemNetDictionaries`)

Usage

```
best.guess(word, dictionary, tolerance = 1)
```

Arguments

`word`	Character. A word to get best guess spelling options from dictionary
`dictionary`	Character vector. The dictionary to search for best guesses in. See `SemNetDictionaries`
`tolerance`	Numeric. The distance tolerance set for automatic spell-correction purposes. This function uses the function `stringdist` to compute the Damerau-Levenshtein distance, which is used to determine potential best guesses Unique words (i.e., $n = 1$) that are within the (distance) tolerance are automatically output as best guess responses, which are then passed through `word.check.wrapper`. This default is based on Damerau's (1964) proclamation that more than 80 (e.g., insertion, deletion, substitution, and transposition). If there is more than one word that is within or below the distance tolerance, then these will be provided as potential options. The recommended and default distance tolerance is `tolerance = 1`, which only spell corrects a word if there is only one word with a DL distance of 1.

Value

The best guess(es) of the word

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*, 171-176.

Examples

```
# Misspelled "bombay"
best.guess("bomba", SemNetDictionaries::animals.dictionary)
```

bin2resp *Binary Responses to Character Responses*

Description

Converts the binary response matrix into characters for each participant

Usage

```
bin2resp(rmat, to.data.frame = FALSE)
```

Arguments

rmat	Binary matrix. A binarized response matrix of verbal fluency or linguistic data
to.data.frame	Boolean. Should output be a data frame where participants are columns? Defaults to FALSE. Set to TRUE to convert output to data frame

Value

A list containing objects for each participant and their responses

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")

# Change binary response matrix to word response matrix
charmat <- bin2resp(clean$binary)
```

combine.responses *Combine Words Wrapper*

Description

A wrapper function to combine words that are found in dictionary (e.g., "star" "fish" –> "starfish")

Usage

```
combine.responses(vec, dictionary)
```

Arguments

vec	Vector. A vector with words to potentially be combined
dictionary	A dictionary to look for (combined) word in. See SemNetDictionaries

Value

A vector with responses combined based on dictionary entries

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Convert "star fish" to "starfish"
combine.responses("star fish", SemNetDictionaries::animals.dictionary)
```

---

converge *Converge Responses*

Description

Merge a column of binarized response data with another

Usage

```
converge(rmat, word, replace)
```

Arguments

rmat	Binary response matrix. A textcleaner filtered response matrix
word	Character. Must be column name (characters). The column name (or number) that be merged *into*. This column will *remain* in the matrix
replace	Character. Must be column name (characters). The column name (or number) that should be merged with the word column. This column will be *removed* from the matrix

Value

The response matrix with the word column merged and the replace column removed

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")

# Converge "golden retriever" into response of "lab"
rmat <- converge(clean$binary,"lab","golden retriever")

# "lab" remains will "golden retriever" responses are merged into
# "lab" and "golden retriever" is removed
```

`correct.changes` *Correct Changes from* `textcleaner`

Description

Allows corrections to changes made by `textcleaner`. Some changes may have been made by accident, some changes may have been made by the automated cleaning, while others may just need to be removed. This function will correct any changes made in a cleaned `textcleaner` object.

Usage

```
correct.changes(textcleaner.obj, dictionary = NULL, old)
```

Arguments

`textcleaner.obj`	A `textcleaner` object
`dictionary`	Character vector. Can be a vector of a corpus or any text for comparison. Dictionary to be used for more efficient text cleaning. Defaults to `NULL`, which will use `general.dictionary`
`old`	Character vector. A vector of old response(s) to change. See the object `spellcheck$auto` in `textcleaner` output

Details

This function is used to correct mistakes that occur in the cleaning process during `textcleaner`. There are times when you are too deep into the text cleaning process that accidentally hitting a '1' instead of a '2' does not make sense to stop and start the text cleaning process over. Rather when mistakes are made, a record can be kept and this function will allow those mistakes to be amended.

Old responses should be used as input. A menu will prompt the user for their decision on how to manage the incorrectly cleaned response. There are three potential options:

- `1: TYPE MY OWN` Allows user to type their own response. If multiple responses, then commas should separate each response. Quotations are not necessary.

- `2: GOOGLE IT` "Googles" the response in question. A browser will open with the Google search terms: define "RESPONSE"
- `3: BAD RESPONSE` When selected, `NA` will be returned

Value

This function returns a list containing the following `textcleaner` objects, which have been corrected with the user-provided changes:

`binary` A matrix of responses where each row represents a participant and each column represents a unique response. A response that a participant has provided is a '`1`' and a response that a participant has not provided is a '`0`'

`resposnes` A response matrix that has been spell-checked and de-pluralized with duplicates removed. This can be used as a final dataset for analyses (e.g., fluency of responses)

`spellcheck` A list containing three objects:

- `full` All responses regardless of spell-checking changes
- `unique` Only responses that were changed during spell-check (includes correct responses that were changed to singular form and lower case)
- `auto` Only the incorrect responses that were changed during spell-check

`removed` A list containing two objects:

- `rows` Identifies removed participants by their row (or column) location in the original data file
- `ids` Identifies removed participants by their ID (see argument `data`

`partChanges` A list where each participant is a list index with each response that was been changed. Participants are identified by their ID (see argument `data`). This can be used to replicate the cleaning process and to keep track of changes more generally. Participants with `NA` did not have any changes from their original data and participants with missing data are removed (see `removed$ids`)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")

# Correct changes
if(interactive())
{corr.clean <- correct.changes(clean, old = "rat", dictionary = "animals")}
```

destr *De-string Responses*

Description

De-string responses after performing textcleaner

Usage

```
destr(rmat, column, sep)
```

Arguments

rmat	A textcleaner filtered response matrix
column	The column number or name of the stringed response
sep	Separating string character (e.g., " ", ".", ","). Must be input as a character

Value

A list containing four objects:

rmat	A response matrix that has been de-stringed
part	The row number is supplied for each case that was affected. This can be used to replicate the de-stringing process and to keep track of changes more generally
added	Stringed responses that were added to the response matrix
removed	Stringed responses that were removed from the response matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")

# Obtain binary data
bin <- clean$binary

# Change column name as an example
colnames(bin)[1] <- "alpaca.ant.antelope"

# De-string
if(interactive())
{convmat <- destr(bin, "alpaca.ant.antelope", ".")}
```

dup.match *Detect Duplicate Matches*

Description

A wrapper function for correct.changes. It returns the opposite values of what the name of the function suggests – that is, FALSE for duplicates and TRUE for non-duplicates

Usage

```
dup.match(tc.obj, part, target)
```

Arguments

tc.obj	A textcleaner object
part	Participant ID
target	Target response for correcting the change (see old argument in correct.changes)

Value

Returns FALSE for responses that have been identified twice in either the participant's original responses (i.e., tc.obj$responses$orig) or in their changed responses (i.e., tc.obj$partChanges). Returns TRUE if response is not given

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")

# Check for duplicate match
dup.match(clean, 1, 1)
```

`equate` *Equate Groups*

Description

A function to "equate" to multiple response matrices. *N* number of groups are matched based on their responses so that every group has the same responses in their data

Usage

```
equate(...)
```

Arguments

`...`	Matrices or data frames. Binary response matrices to be equated

Value

This function returns a list containing the equated binary response matrices in the order they were input. The response matrices are labeled as the object name they were entered with

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")

# Obtain binary data
bin <- clean$binary

# Finalize mat1
mat1 <- finalize(bin[c(1:5),])

# Finalize mat2
mat2 <- finalize(bin[c(6:10),])


# Equate mat1 and mat1
eq <- equate(mat1, mat2)

# Obtain respective equated response matrices
eq.mat1 <- eq$mat1
eq.mat2 <- eq$mat2
```

`finalize` *Finalize Response Matrix*

Description

Finalizes the response matrix by keeping responses that are given by two or more people

Usage

```
finalize(rmat, minCase = 2)
```

Arguments

`rmat`	Binary matrix. A textcleaner filtered response matrix
`minCase`	Numeric. Minimum number of cases to produce a response

Value

A binary response matrix with responses given by at least `minCase` people

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")

# Obtain binary data
bin <- clean$binary

# Finalize mat1
mat1 <- finalize(bin)
```

full.match *Wrapper Match Function*

Description

A wrapper function that performs the same operations as match except accounts for NA

Usage

```
full.match(vec1, vec2)
```

Arguments

vec1	Vector. Must be same length as vec2
vec2	Vector. Must be same length as vec1

Value

Returns a vector the same length as the input vectors of TRUE and FALSE for each element across the vectors

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Vector 1
vec1 <- c(NA,NA,"cat","dog",NA,0,"porcupine")

# Vector 2
vec2 <- c(NA,"bob","alice","dog","prince",0,NA)

# Perform match
full.match(vec1, vec2)
```

is.letter *Checks If A Character Is A Letter*

Description

A wrapper function designed to determine whether a single character is a letter

Usage

```
is.letter(letter)
```

Arguments

letter	A single character

Value

A TRUE or FALSE value for whether the character entered is a letter

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# TRUE
is.letter("r")

# FALSE
is.letter("5")

# FALSE
is.letter("~")
```

---

lead.word *Lead Word Example*

Description

An example word demonstrating that trimws does not remove leading space for a word

Usage

```
data(lead.word)
```

Format

lead.word (vector, length = 1)

Examples

```
data("lead.word")
```

moniker *Moniker Function*

Description

A wrapper function for spell-checking (identifies monikers for a word)

Usage

```
moniker(word, misnom)
```

Arguments

word	Word to check for moniker
misnom	A list of monikers. See dictionaries for options

Value

If word matches a moniker, then the appropriate word is returned. If word does not match a moniker, then the word is returned

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
moniker("possum", SemNetDictionaries::animals.moniker)
```

multi.word.check *Multi Word Checker Wrapper*

Description

A wrapper function to spell-check responses that have more than one word in them

Usage

```
multi.word.check(string, dictionary, tolerance)
```

Arguments

`string`	Character. A string of words with a `length = 1`
`dictionary`	A dictionary to look for word in (see examples). See `SemNetDictionaries`
`tolerance`	Numeric. The distance tolerance set for automatic spell-correction purposes. This function uses the function `stringdist` to compute the Damerau-Levenshtein (DL) distance, which is used to determine potential best guesses. Unique words (i.e., $n = 1$) that are within the (distance) tolerance are automatically output as best guess responses, which are then passed through `word.check.wrapper`. If there is more than one word that is within or below the distance tolerance, then these will be provided as potential options. The recommended and default distance tolerance is `tolerance = 1`, which only spell corrects a word if there is only one word with a DL distance of 1.

Value

Either a spell-corrected response or the original response

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Returns "guinea pig"
multi.word.check("guinea big", SemNetDictionaries::animals.dictionary, tolerance = 1)

# Returns original response
multi.word.check("cat dog bear fish bull", SemNetDictionaries::animals.dictionary, tolerance = 1)
```

`open.animals` *Openness and Verbal Fluency*

Description

Raw Animals verbal fluency data ($n = 516$) from Christensen et al. (2018).

Usage

```
data(open.animals)
```

Format

open.animals (matrix 516 x 38)

Details

First column is a grouping variable (`"Group"`) with `1` corresponding to low openness to experience and `2` to high openness to experience

Second column is the latent variable of openness to experience with Intellect items removed (see Christensen et al., 2018 for more details).

Third column is the ID variable for each participant.

Columns 4-38 are raw fluency data.

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, *32*, 480-492. doi:10.1002/per.2157

Examples

```
data("open.animals")
```

`pluralize` *Converts Words to their Plural Form*

Description

A function to change words to their plural form. The rules for converting words to their plural forms are based on the grammar rules found here: https://www.grammarly.com/blog/plural-nouns/. This function does not handle special cases, so particular care is necessary.

Usage

```
pluralize(word)
```

Arguments

`word`	A word

Value

Returns the word in singular form

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Handles any prototypical cases
"dogs"
pluralize("dog")

"foxes"
pluralize("fox")

"wolves"
pluralize("wolf")

"octopi"
pluralize("octopus")

"taxa"
pluralize("taxon")

# And most special cases:
"wives"
pluralize("wife")

"roofs"
pluralize("roof")

"photos"
pluralize("photo")

# And some irregular cases:
"children"
pluralize("child")

"teeth"
pluralize("tooth")

"mice"
pluralize("mouse")
```

---

qwerty.dist *QWERTY Distance for Same Length Words*

Description

Computes QWERTY Distance for words that have the same number of characters. Distance is computed based on the number of keys a character is away from another character on a QWERTY keyboard

Usage

```
qwerty.dist(wordA, wordB)
```

Arguments

wordA	Character vector. Word to be compared
wordB	Character vector. Word to be compared

Value

Numeric value for distance between wordA and wordB

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
#Identical values for Damerau-Levenshtein
stringdist::stringdist("big", "pig", method="dl")

stringdist::stringdist("big", "bug", method="dl")

#Different distances for QWERTY
qwerty.dist("big", "pig")

qwerty.dist("big", "bug") # Probably meant to type "bug"
```

---

read.data *Read in Common Data File Extensions*

Description

A single function to read in common data file extensions

File extensions supported:

- .Rdata
- .rds
- .csv
- .xlsx
- .xls
- .sav
- .txt
- .mat

Usage

```
read.data(file = file.choose(), header = TRUE, sep = ",", ...)
```

Arguments

file	Character. A path to the file to load. Defaults to interactive file selection using file.choose
header	Boolean. A logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: header is set to TRUE if and only if the first row contains one fewer field than the number of columns
sep	Character. The field separator character. Values on each line of the file are separated by this character. If sep = "" (the default for read.table) the separator is a 'white space', that is one or more spaces, tabs, newlines or carriage returns
...	Additional arguments. Allows for additional arguments to be passed onto the respective read functions. See documentation in the list below: • .Rdata load • .rds readRDS • .csv read.table • .xlsx read_excel • .xls read_excel • .sav read.spss • .txt read.table • .mat readMat

Value

A data frame containing a representation of the data in the file. If file extension is ".Rdata", then data will be read to the global environment

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

R Core Team R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

readxl Hadley Wickham and Jennifer Bryan (2019). readxl: Read Excel Files. R package version 1.3.1. https://CRAN.R-project.org/package=readxl

R.matlab Henrik Bengtsson (2018). R.matlab: Read and Write MAT Files and Call MATLAB from Within R. R package version 3.6.2. https://CRAN.R-project.org/package=R.matlab

Examples

```
# Use this example for your data
if(interactive())
{read.data()}

# Example for CRAN tests
```

```
## Create test data
test1 <- c(1:5, "6,7", "8,9,10")

## Path to temporary file
tf <- tempfile()

## Create test file
writeLines(test1, tf)

## Read in data
read.data(tf)

# See documentation of respective R functions for specific examples
```

`rm.lead.space` *Removes Leading Spaces*

Description

Removes leading spaces that are not caught by `trimws`

Usage

```
rm.lead.space(word)
```

Arguments

`word`	Character (vector). A word that has leading spaces that cannot be removed by `trimws`

Value

Word without leading spaces

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# 'trimws' should remove lead space but doesn't
trimws(lead.word)

# 'rm.lead.space' does
rm.lead.space(lead.word)
```

`singularize` *Converts Words to their Singular Form*

Description

A function to change words to their singular form. The rules for converting words to their singular forms are based on the ***opposite*** of the grammar rules found here: https://www.grammarly.com/blog/plural-nouns/. This function does not handle special cases, so particular care is necessary.

Usage

```
singularize(word)
```

Arguments

`word`	A word

Value

Returns the word in singular form

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Handles any prototypical cases
"dog"
singularize("dogs")

"fox"
singularize("foxes")

"wolf"
singularize("wolves")

"octopus"
singularize("octopi")

"taxon"
singularize("taxa")

# And most special cases:
"wife"
singularize("wives")

"fez"
singularize("fezzes")
```

```
"roof"
singularize("roofs")

"photo"
singularize("photos")

# And some irregular cases:
"child"
singularize("children")

"tooth"
singularize("teeth")

"mouse"
singularize("mice")
```

`spell.check.dictionary`

Spelling-check using SemNetDictionaries

Description

A wrapper function for spell-checking text dictionaries in `SemNetDictionaries` (combines all spell-checking wrapper functions)

Usage

```
spell.check.dictionary(check, dictionary, part.resp, tolerance = 1)
```

Arguments

`check`	Character vector. A vector of unique responses from text data
`dictionary`	Character vector. See `SemNetDictionaries`
`part.resp`	Matrix or data frame. Uncleaned participant response matrix
`tolerance`	Numeric. The distance tolerance set for automatic spell-correction purposes. This function uses the function `stringdist` to compute the Damerau-Levenshtein (DL) distance, which is used to determine potential best guesses. Unique words (i.e., $n = 1$) that are within the (distance) tolerance are automatically output as best guess responses, which are then passed through `word.check.wrapper`. If there is more than one word that is within or below the distance tolerance, then these will be provided as potential options. The recommended and default distance tolerance is `tolerance = 1`, which only spell corrects a word if there is only one word with a DL distance of 1.

Value

Returns a list containing:

`from`	A list of all responses before they were cleaned
`to`	A list of all responses after they were cleaned
`dict`	The updated dictionary vector
`from.inc`	A list of only incorrect responses before they were cleaned
`to.inc`	A list of only incorrect responses after they were cleaned

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

if(interactive())
{
    scd <- spell.check.dictionary(check = unique(unlist(raw)),
    dictionary = SemNetDictionaries::animals.dictionary,
    part.resp = raw)
}
```

---

splitstr.check *Split String Check*

Description

A wrapper function for spell-checking (ensures next word does not belong to the previous)

Usage

```
splitstr.check(string, split = " ", dictionary, remember = list())
```

Arguments

`string`	Character. A string of words (see examples)
`split`	Character. A character that should be used to "split" the words input into the `string` argument. Defaults to a space (`" "`)
`dictionary`	Dictionary to check. See `SemNetDictionaries`
`remember`	Character list. Checks if split string has already been checked (a wrapper argument for `spell.check.dictionary`). Defaults to an empty list

Value

Returns the string as is or with the selected responses merged

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Create long word vector
words <- "bombay opossum guinea pig horse cow"

if(interactive())
{splitstr.check(string = words, split = " ", dictionary = SemNetDictionaries::animals.dictionary)}
```

---

starting.letter *Starting Letter*

Description

A wrapper function designed to produce the first letter that appears in a word, regardless of leading characters

Usage

```
starting.letter(word)
```

Arguments

word	Character. A single word

Value

First letter in the string

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# First letter is "w"
starting.letter("..walrus")

# First letter is "r"
starting.letter("5rat")

# First letter is "b"
```

```
starting.letter("%1.,bombay")
```

`textcleaner` *Text Cleaner*

Description

An automated cleaning function for spell-checking, de-pluralizing, removing duplicates, and binarizing text data

Usage

```
textcleaner(data, miss = 99, partBY = c("row", "col"),
  dictionary = NULL, tolerance = 1)
```

Arguments

`data`	Matrix or data frame. A dataset of text data. Participant IDs should be made to be row or column names to specify whether participants are by row or column (see argument `partBY`). If no IDs are provided, then their order in the corresponding row (or column is used). A message will notify the user how IDs were assigned
`miss`	Numeric or character. Value for missing data. Defaults to `99`
`partBY`	Character. Are participants by row or column? Set to `"row"` for by row. Set to `"col"` for by column
`dictionary`	Character vector. Can be a vector of a corpus or any text for comparison. Dictionary to be used for more efficient text cleaning. Defaults to `NULL`, which will use `general.dictionary` Use `dictionaries()` or `find.dictionaries()` for more options (See `SemNetDictionaries` for more details)
`tolerance`	Numeric. The distance tolerance set for automatic spell-correction purposes. This function uses the function `stringdist` to compute the Damerau-Levenshtein (DL) distance, which is used to determine potential best guesses. Unique words (i.e., $n = 1$) that are within the (distance) tolerance are automatically output as best guess responses, which are then passed through `word.check.wrapper`. If there is more than one word that is within or below the distance tolerance, then these will be provided as potential options. The recommended and default distance tolerance is `tolerance = 1`, which only spell corrects a word if there is only one word with a DL distance of 1.

Details

When working through the menu options in textcleaner, there may be mistakes. For instance, selecting to REMOVE a response when really all you wanted to do was RENAME a response. There are a couple of options:

RECOMMENDED

1. You can make a note in your R script for the change you wanted to make (you can keep moving through the cleaning process). After the cleaning process is through, you can check the spellcheck$unique output of textcleaner to see what changes you made. To correct any changes you made in the cleaning process, you can use the correct.changes function

NOT RECOMMENDED

2. You can use esc to exit out of a menu selection process. This is NOT recommended because you will lose all changes that you've made up to that point

Value

This function returns a list containing the following objects:

`binary` A matrix of responses where each row represents a participant and each column represents a unique response. A response that a participant has provided is a '1' and a response that a participant has not provided is a '0'

`responses` A list containing two objects:

- clean.respA response matrix that has been spell-checked and de-pluralized with duplicates removed. This can be used as a final dataset for analyses (e.g., fluency of responses)
- orig.respThe original response matrix that has had white spaces before and after words response. Also converts all upper-case letters to lower case

`spellcheck` A list containing three objects:

- `full` All responses regardless of spell-checking changes
- `unique` Only responses that were changed during spell-check (includes correct responses that were changed to singular form and lower case)
- `auto` Only the incorrect responses that were changed during spell-check

`removed` A list containing two objects:

- `rows` Identifies removed participants by their row (or column) location in the original data file
- `ids` Identifies removed participants by their ID (see argument `data`

`partChanges` A list where each participant is a list index with each response that was been changed. Participants are identified by their ID (see argument `data`). This can be used to replicate the cleaning process and to keep track of changes more generally. Participants with NA did not have any changes from their original data and participants with missing data are removed (see `removed$ids`)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Hornik, K., & Murdoch, D. (2010). Watch Your Spelling!. *The R Journal*, *3*, 22-28. doi:10.32614/RJ-2011-014

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

# Clean and prepocess data
clean <- textcleaner(raw, partBY = "row", dictionary = "animals")
if(interactive())
{
    #Full test
    clean <- textcleaner(open.animals[,-c(1,2)], partBY = "row", dictionary = "animals")
}
```

`word.check.wrapper` *A Spell-checking wrapper*

Description

A wrapper function to spell-check with menu options

Usage

```
word.check.wrapper(word, dictionary, context = NULL, part.resp,
  tolerance = 1, rem.resp)
```

Arguments

`word`	Character. A word to get spell-checked
`dictionary`	A dictionary to look for word in (see examples). See `SemNetDictionaries`
`context`	Vector. Defaults to `NULL`. When a word is inside of a vector of words, then the vector can be input to provide context for whether this word is spelled correctly with other words. For example, "guinea" is spelled correctly but will not be in `animals.dictionary`. The vector can be input to determine if "guinea pig" or "guinea fowl" is meant by the participant. The word that is being checked will appear with "«" and "»" around it in the context of other words (e.g., `bat dog fish <<guinea>> pig rat horse`)
`part.resp`	Matrix or data frame. Uncleaned participant response matrix
`tolerance`	Numeric. The distance tolerance set for automatic spell-correction purposes. This function uses the function `stringdist` to compute the Damerau-Levenshtein (DL) distance, which is used to determine potential best guesses. Unique words (i.e., $n = 1$) that are within the (distance) tolerance are automatically output as best guess responses, which are then passed through `word.check.wrapper`.

	If there is more than one word that is within or below the distance tolerance, then these will be provided as potential options. The recommended and default distance tolerance is `tolerance = 1`, which only spell corrects a word if there is only one word with a DL distance of 1.
`rem.resp`	Matrix. Keeps track of decisions made in the cleaning process

Details

A menu will appear with several options. Here is what is returned with each option:

- POTENTIAL RESPONSEIf a potential response is selected, then the input word is replaced with the potential response
- ADD TO DICTIONARYWhen selected, the input word will be added to the appendix dictionary (see `append.dictionary`). The input word will be returned
- TYPE MY OWNUser will type their own response to replace the input word. If word is not in dictionary, then user will be prompted for whether they would like to add the word to their appendix dictionary (see `append.dictionary`). In all cases, the typed word will be returned
- CONTEXTProvides the response in context of the participant's other responses. Prints the all participants responses that were given with the target response
- GOOGLE IT"Googles" the response in question. A browser will open with the Google search terms: define "RESPONSE"
- BAD RESPONSEWhen selected, `NA` will be returned
- SKIPWhen selected, input word will be returned
- BAD STRINGUnique to continuous strings. When selected, a vector of `NA` the length of the `context` vector will be returned
- CONTEXTUnique to single responses. Provides the response in context of the participant's other responses. Prints the all participants responses that were given with the target response

Value

A list containing:

`word`	The spelling corrected word
`dictionary`	The updated dictionary
`check`	A check for whether a word has been added to the dictionary
`rem.resp`	A matrix to remember previous responses for spelling corrections

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
if(interactive())
{
    # Response needs to be checked
    word.check.wrapper("gost", SemNetDictionaries::animals.dictionary)
}
```

Index