Package: PsychWordVec (via r-universe)

August 26, 2024

Title Word Embedding Research Framework for Psychological Science

Version 2023.9

Date 2023-09-27

Maintainer Han-Wu-Shuang Bao <baohws@foxmail.com>

Description An integrative toolbox of word embedding research that provides: (1) a collection of 'pre-trained' static word vectors in the '.RData' compressed format <https://psychbruce.github.io/WordVector_RData.pdf>; (2) a series of functions to process, analyze, and visualize word vectors; (3) a range of tests to examine conceptual associations, including the Word Embedding Association Test <doi:10.1126/science.aal4230> and the Relative Norm Distance <doi:10.1073/pnas.1720347115>, with permutation test of significance; (4) a set of training methods to locally train (static) word vectors from text corpora, including 'Word2Vec' <arXiv:1301.3781>, 'GloVe' <doi:10.3115/v1/D14-1162>, and 'FastText' <arXiv:1607.04606>; (5) a group of functions to download 'pre-trained' language models (e.g., 'GPT', 'BERT') and extract contextualized (dynamic) word vectors (based on the R package 'text').

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

URL https://psychbruce.github.io/PsychWordVec/

BugReports https://github.com/psychbruce/PsychWordVec/issues

Depends R (>= 4.0.0)

Imports bruceR, dplyr, stringr, data.table, purrr, vroom, cli, ggplot2, ggrepel, corrplot, psych, Rtsne, rgl, qgraph, rsparse, text2vec, word2vec, fastTextR, text, reticulate

Suggests wordsalad, sweater, glue

RoxygenNote 7.2.3

Repository https://psychbruce.r-universe.dev

RemoteUrl https://github.com/psychbruce/psychwordvec

RemoteRef HEAD

RemoteSha 0f1b53e2fbd7e7ca3696aab28df0da974a66f16b

Contents

as_embed *Word vectors data class:* wordvec *and* embed.

Description

PsychWordVec uses two types of word vectors data: wordvec (data.table, with two variables word and vec) and embed (matrix, with dimensions as columns and words as row names). Note that matrix operation makes embed much faster than wordvec. Users are suggested to reshape data to embed before using the other functions.

Usage

```
as_embed(x, normalize = FALSE)

as_wordvec(x, normalize = FALSE)

## S3 method for class 'embed'
x[i, j]

pattern(pattern)
```

Arguments

x	Object to be reshaped. See examples.
normalize	Normalize all word vectors to unit length? Defaults to FALSE. See normalize.
i, j	Row (i) and column (j) filter to be used in embed[i, j].
pattern	Regular expression to be used in embed[pattern("...")].

Value

A wordvec (data.table) or embed (matrix).

Functions

- as_embed(): From wordvec (data.table) to embed (matrix).
- as_wordvec(): From embed (matrix) to wordvec (data.table).

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

load_wordvec / load_embed

normalize

data_transform

data_wordvec_subset

Examples

```
dt = head(demodata, 10)
str(dt)

embed = as_embed(dt, normalize=TRUE)
embed
str(embed)

wordvec = as_wordvec(embed, normalize=TRUE)
wordvec
str(wordvec)

df = data.frame(token=LETTERS, D1=1:26/10000, D2=26:1/10000)
as_embed(df)
as_wordvec(df)

dd = rbind(dt[1:5], dt[1:5])
dd  # duplicate words
unique(dd)

dm = as_embed(dd)
dm  # duplicate words
unique(dm)

# more examples for extracting a subset using `x[i, j]`
# (3x faster than `wordvec`)
embed = as_embed(demodata)
embed[1]
embed[1:5]
embed["for"]
embed[pattern("^for.{0,2}$")]
embed[cc("for, in, on, xxx")]
embed[cc("for, in, on, xxx"), 5:10]
embed[1:5, 5:10]
embed[, 5:10]
embed[3, 4]
embed["that", 4]
```

`cosine_similarity` *Cosine similarity/distance between two vectors.*

Description

Cosine similarity/distance between two vectors.

Usage

```
cosine_similarity(v1, v2, distance = FALSE)

cos_sim(v1, v2)

cos_dist(v1, v2)
```

Arguments

`v1, v2`	Numeric vector (of the same length).
`distance`	Compute cosine distance instead? Defaults to `FALSE` (cosine similarity).

Details

Cosine similarity =

`sum(v1 * v2) / ( sqrt(sum(v1^2)) * sqrt(sum(v2^2)) )`

Cosine distance =

`1 - cosine_similarity(v1, v2)`

Value

A value of cosine similarity/distance.

See Also

`pair_similarity`

`tab_similarity`

`most_similar`

Examples

```
cos_sim(v1=c(1,1,1), v2=c(2,2,2))  # 1
cos_sim(v1=c(1,4,1), v2=c(4,1,1))  # 0.5
cos_sim(v1=c(1,1,0), v2=c(0,0,1))  # 0

cos_dist(v1=c(1,1,1), v2=c(2,2,2))  # 0
cos_dist(v1=c(1,4,1), v2=c(4,1,1))  # 0.5
cos_dist(v1=c(1,1,0), v2=c(0,0,1))  # 1
```

data_transform *Transform plain text of word vectors into* wordvec *(data.table) or* embed *(matrix), saved in a compressed ".RData" file.*

Description

Transform plain text of word vectors into wordvec (data.table) or embed (matrix), saved in a compressed ".RData" file.

Speed: In total (preprocess + compress + save), it can process about 30000 words/min with the slowest settings (compress="xz", compress.level=9) on a modern computer (HP ProBook 450, Windows 11, Intel i7-1165G7 CPU, 32GB RAM).

Usage

```
data_transform(
  file.load,
  file.save,
  as = c("wordvec", "embed"),
  sep = " ",
  header = "auto",
  encoding = "auto",
  compress = "bzip2",
  compress.level = 9,
  verbose = TRUE
)
```

Arguments

file.load	File name of raw text (must be plain text). Data must be in this format (values separated by sep): cat 0.001 0.002 0.003 0.004 0.005 ... 0.300 dog 0.301 0.302 0.303 0.304 0.305 ... 0.600
file.save	File name of to-be-saved R data (must be .RData).
as	Transform the text to which R object? wordvec (data.table) or embed (matrix). Defaults to wordvec.
sep	Column separator. Defaults to " ".
header	Is the 1st row a header (e.g., meta-information such as "2000000 300")? Defaults to "auto", which automatically determines whether there is a header. If TRUE, then the 1st row will be dropped.
encoding	File encoding. Defaults to "auto" (using vroom::vroom_lines() to fast read the file). If specified to any other value (e.g., "UTF-8"), then it uses readLines() to read the file, which is much slower than vroom.
compress	Compression method for the saved file. Defaults to "bzip2". Options include:

- 1 or "gzip": modest file size (fastest)
- 2 or "bzip2": small file size (fast)
- 3 or "xz": minimized file size (slow)

compress.level Compression level from 0 (none) to 9 (maximal compression for minimal file size). Defaults to 9.

verbose Print information to the console? Defaults to TRUE.

Value

A wordvec (data.table) or embed (matrix).

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

as_wordvec / as_embed

load_wordvec / load_embed

normalize

data_wordvec_subset

Examples

```
## Not run:
# please first manually download plain text data of word vectors
# e.g., from: https://fasttext.cc/docs/en/crawl-vectors.html

# the text file must be on your disk
# the following code cannot run unless you have the file
library(bruceR)
set.wd()
data_transform(file.load="cc.zh.300.vec",   # plain text file
               file.save="cc.zh.300.vec.RData",  # RData file
               header=TRUE, compress="xz")  # of minimal size

## End(Not run)
```

data_wordvec_load *Load word vectors data (*wordvec *or* embed*) from ".RData" file.*

Description

Load word vectors data (wordvec or embed) from ".RData" file.

Usage

```
data_wordvec_load(
  file,
  as = c("wordvec", "embed"),
  normalize = FALSE,
  verbose = TRUE
)

load_wordvec(file, normalize = TRUE)

load_embed(file, normalize = TRUE)
```

Arguments

file	File name of .RData transformed by data_transform. Can also be an .RData file containing an embedding matrix with words as row names.
as	Load as wordvec (data.table) or embed (matrix). Defaults to the original class of the R object in file. The two wrapper functions load_wordvec and load_embed automatically reshape the data to the corresponding class and normalize all word vectors (for faster future use).
normalize	Normalize all word vectors to unit length? Defaults to FALSE. See normalize.
verbose	Print information to the console? Defaults to TRUE.

Value

A wordvec (data.table) or embed (matrix).

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

as_wordvec / as_embed

normalize

data_transform

data_wordvec_subset

Examples

```
d = demodata[1:200]
save(d, file="demo.RData")
d = load_wordvec("demo.RData")
d
d = load_embed("demo.RData")
d
unlink("demo.RData")  # delete file for code check

## Not run:
# please first manually download the .RData file
# (see https://psychbruce.github.io/WordVector_RData.pdf)
# or transform plain text data by using `data_transform()`

# the RData file must be on your disk
# the following code cannot run unless you have the file
library(bruceR)
set.wd()
d = load_embed("../data-raw/GloVe/glove_wiki_50d.RData")
d

## End(Not run)
```

`data_wordvec_subset` *Extract a subset of word vectors data (with S3 methods).*

Description

Extract a subset of word vectors data (with S3 methods). You may specify either a `wordvec` or `embed` loaded by `data_wordvec_load`) or an .RData file transformed by `data_transform`).

Usage

```
data_wordvec_subset(
  x,
  words = NULL,
  pattern = NULL,
  as = c("wordvec", "embed"),
  file.save,
  compress = "bzip2",
  compress.level = 9,
  verbose = TRUE
)

## S3 method for class 'wordvec'
subset(x, ...)
```

```
## S3 method for class 'embed'
subset(x, ...)
```

Arguments

x	Can be: • a wordvec or embed loaded by data_wordvec_load • an .RData file transformed by data_transform
words	[Option 1] Character string(s).
pattern	[Option 2] Regular expression (see str_subset). If neither words nor pattern are specified (i.e., both are NULL), then all words in the data will be extracted.
as	Reshape to wordvec (data.table) or embed (matrix). Defaults to the original class of x.
file.save	File name of to-be-saved R data (must be .RData).
compress	Compression method for the saved file. Defaults to "bzip2". Options include: • 1 or "gzip": modest file size (fastest) • 2 or "bzip2": small file size (fast) • 3 or "xz": minimized file size (slow)
compress.level	Compression level from 0 (none) to 9 (maximal compression for minimal file size). Defaults to 9.
verbose	Print information to the console? Defaults to TRUE.
...	Parameters passed to data_wordvec_subset when using the S3 method subset.

Value

A subset of wordvec or embed of valid (available) words.

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

as_wordvec / as_embed

load_wordvec / load_embed

get_wordvec

data_transform

Examples

```
## directly use `embed[i, j]` (3x faster than `wordvec`):
d = as_embed(demodata)
d[1:5]
d["people"]
d[c("China", "Japan", "Korea")]

## specify `x` as a `wordvec` or `embed` object:
subset(demodata, c("China", "Japan", "Korea"))
subset(d, pattern="^Chi")

## specify `x` and `pattern`, and save with `file.save`:
subset(demodata, pattern="Chin[ae]|Japan|Korea",
       file.save="subset.RData")

## load the subset:
d.subset = load_wordvec("subset.RData")
d.subset

## specify `x` as an .RData file and save with `file.save`:
data_wordvec_subset("subset.RData",
                    words=c("China", "Chinese"),
                    file.save="new.subset.RData")
d.new.subset = load_embed("new.subset.RData")
d.new.subset

unlink("subset.RData")  # delete file for code check
unlink("new.subset.RData")  # delete file for code check
```

`demodata` *Demo data (pre-trained using word2vec on Google News; 8000 vocab, 300 dims).*

Description

This demo data contains a sample of 8000 English words with 300-dimension word vectors pre-trained using the "word2vec" algorithm based on the Google News corpus. Most of these words are from the Top 8000 frequent wordlist, whereas a few are selected from less frequent words and appended.

Usage

```
data(demodata)
```

Format

A `data.table` (of new class `wordvec`) with two variables `word` and `vec`, transformed from the raw data (see the URL in Source) into `.RData` using the `data_transform` function.

Source

Google Code - word2vec (https://code.google.com/archive/p/word2vec/)

Examples

```
class(demodata)
demodata

embed = as_embed(demodata, normalize=TRUE)
class(embed)
embed
```

`dict_expand` *Expand a dictionary from the most similar words.*

Description

Expand a dictionary from the most similar words.

Usage

```
dict_expand(data, words, threshold = 0.5, iteration = 5, verbose = TRUE)
```

Arguments

`data`	A `wordvec` (data.table) or `embed` (matrix), see `data_wordvec_load`.
`words`	A single word or a list of words, used to calculate the sum vector.
`threshold`	Threshold of cosine similarity, used to find all words with similarities higher than this value. Defaults to `0.5`. A low threshold may lead to failure of convergence.
`iteration`	Number of maximum iterations. Defaults to `5`.
`verbose`	Print information to the console? Defaults to `TRUE`.

Value

An expanded list (character vector) of words.

Download

Download pre-trained word vectors data (`.RData`): https://psychbruce.github.io/WordVector_RData.pdf

See Also

`sum_wordvec`

`most_similar`

`dict_reliability`

Examples

```
dict = dict_expand(demodata, "king")
dict

dict = dict_expand(demodata, cc("king, queen"))
dict

most_similar(demodata, dict)

dict.cn = dict_expand(demodata, "China")
dict.cn  # too inclusive if setting threshold = 0.5

dict.cn = dict_expand(demodata,
                      cc("China, Chinese"),
                      threshold=0.6)
dict.cn  # adequate to represent "China"
```

`dict_reliability` *Reliability analysis and PCA of a dictionary.*

Description

Reliability analysis (Cronbach's α and average cosine similarity) and Principal Component Analysis (PCA) of a dictionary, with visualization of cosine similarities between words (ordered by the first principal component loading). Note that Cronbach's α can be misleading when the number of items/words is large.

Usage

```
dict_reliability(
  data,
  words = NULL,
  pattern = NULL,
  alpha = TRUE,
  sort = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

`data`	A `wordvec` (data.table) or `embed` (matrix), see `data_wordvec_load`.
`words`	[Option 1] Character string(s).
`pattern`	[Option 2] Regular expression (see `str_subset`). If neither `words` nor `pattern` are specified (i.e., both are `NULL`), then all words in the data will be extracted.

`alpha`	Estimate the Cronbach's α? Defaults to `TRUE`. Note that this can be *misleading* and *time-consuming* when the number of items/words is large.
`sort`	Sort items by the first principal component loading (PC1)? Defaults to `TRUE`.
`plot`	Visualize the cosine similarities? Defaults to `TRUE`.
`...`	Other parameters passed to `plot_similarity`.

Value

A `list` object of new class `reliability`:

`alpha` Cronbach's α

`eigen` Eigen values from PCA

`pca` PCA (only 1 principal component)

`pca.rotation` PCA with varimax rotation (if potential principal components > 1)

`items` Item statistics

`cos.sim.mat` A matrix of cosine similarities of all word pairs

`cos.sim` Lower triangular part of the matrix of cosine similarities

Download

Download pre-trained word vectors data (`.RData`): https://psychbruce.github.io/WordVector_RData.pdf

References

Nicolas, G., Bai, X., & Fiske, S. T. (2021). Comprehensive stereotype content dictionaries using a semi-automated method. *European Journal of Social Psychology, 51*(1), 178–196.

See Also

`cosine_similarity`

`pair_similarity`

`plot_similarity`

`tab_similarity`

`most_similar`

`dict_expand`

Examples

```
d = as_embed(demodata, normalize=TRUE)

dict = dict_expand(d, "king")
dict_reliability(d, dict)

dict.cn = dict_expand(d, "China", threshold=0.65)
dict_reliability(d, dict.cn)
```

```
dict_reliability(d, c(dict, dict.cn))
# low-loading items should be removed
```

`get_wordvec` *Extract word vector(s).*

Description

Extract word vector(s), using either a list of words or a regular expression.

Usage

```
get_wordvec(
  data,
  words = NULL,
  pattern = NULL,
  plot = FALSE,
  plot.dims = NULL,
  plot.step = 0.05,
  plot.border = "white"
)
```

Arguments

`data`	A `wordvec` (data.table) or `embed` (matrix), see `data_wordvec_load`.
`words`	[Option 1] Character string(s).
`pattern`	[Option 2] Regular expression (see `str_subset`). If neither `words` nor `pattern` are specified (i.e., both are `NULL`), then all words in the data will be extracted.
`plot`	Generate a plot to illustrate the word vectors? Defaults to `FALSE`.
`plot.dims`	Dimensions to be plotted (e.g., `1:100`). Defaults to `NULL` (plot all dimensions).
`plot.step`	Step for value breaks. Defaults to `0.05`.
`plot.border`	Color of tile border. Defaults to `"white"`. To remove the border color, set `plot.border=NA`.

Value

A `data.table` with words as columns and dimensions as rows.

Download

Download pre-trained word vectors data (`.RData`): https://psychbruce.github.io/WordVector_RData.pdf

See Also

data_wordvec_subset

plot_wordvec

plot_wordvec_tSNE

Examples

```
d = as_embed(demodata, normalize=TRUE)

get_wordvec(d, c("China", "Japan", "Korea"))
get_wordvec(d, cc(" China, Japan; Korea "))

## specify `pattern`:
get_wordvec(d, pattern="Chin[ae]|Japan|Korea")

## plot word vectors:
get_wordvec(d, cc("China, Japan, Korea,
                  Mac, Linux, Windows"),
            plot=TRUE, plot.dims=1:100)

## a more complex example:

words = cc("
China
Chinese
Japan
Japanese
good
bad
great
terrible
morning
evening
king
queen
man
woman
he
she
cat
dog
")

dt = get_wordvec(
  d, words,
  plot=TRUE,
  plot.dims=1:100,
  plot.step=0.06)

# if you want to change something:
attr(dt, "ggplot") +
```

```
  scale_fill_viridis_b(n.breaks=10, show.limits=TRUE) +
  theme(legend.key.height=unit(0.1, "npc"))

# or to save the plot:
ggsave(attr(dt, "ggplot"),
       filename="wordvecs.png",
       width=8, height=5, dpi=500)
unlink("wordvecs.png")  # delete file for code check
```

`most_similar` *Find the Top-N most similar words.*

Description

Find the Top-N most similar words, which replicates the results produced by the Python `gensim` module `most_similar()` function. (Exact replication of `gensim` requires the same word vectors data, not the `demodata` used here in examples.)

Usage

```
most_similar(
  data,
  x = NULL,
  topn = 10,
  above = NULL,
  keep = FALSE,
  row.id = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	A `wordvec` (data.table) or `embed` (matrix), see `data_wordvec_load`.

`x`	Can be:
	• `NULL`: use the sum of all word vectors in `data`
	• a single word:
	`"China"`
	• a list of words:
	`c("king", "queen")`
	`cc(" king , queen ; man | woman")`
	• an R formula (`~ xxx`) specifying words that positively and negatively contribute to the similarity (for word analogy):
	`~ boy - he + she`
	`~ king - man + woman`
	`~ Beijing - China + Japan`

`topn`	Top-N most similar words. Defaults to `10`.

above	Defaults to NULL. Can be: • a threshold value to find all words with cosine similarities higher than this value • a critical word to find all words with cosine similarities higher than that with this critical word If both topn and above are specified, above wins.
keep	Keep words specified in x in results? Defaults to FALSE.
row.id	Return the row number of each word? Defaults to TRUE, which may help determine the relative word frequency in some cases.
verbose	Print information to the console? Defaults to TRUE.

Value

A data.table with the most similar words and their cosine similarities.

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

sum_wordvec

dict_expand

dict_reliability

cosine_similarity

pair_similarity

plot_similarity

tab_similarity

Examples

```
d = as_embed(demodata, normalize=TRUE)

most_similar(d)
most_similar(d, "China")
most_similar(d, c("king", "queen"))
most_similar(d, cc(" king , queen ; man | woman "))

# the same as above:
most_similar(d, ~ China)
most_similar(d, ~ king + queen)
most_similar(d, ~ king + queen + man + woman)

most_similar(d, ~ boy - he + she)
most_similar(d, ~ Jack - he + she)
most_similar(d, ~ Rose - she + he)
```

```
most_similar(d, ~ king - man + woman)
most_similar(d, ~ Tokyo - Japan + China)
most_similar(d, ~ Beijing - China + Japan)

most_similar(d, "China", above=0.7)
most_similar(d, "China", above="Shanghai")

# automatically normalized for more accurate results
ms = most_similar(demodata, ~ king - man + woman)
ms
str(ms)
```

`normalize` *Normalize all word vectors to the unit length 1.*

Description

L2-normalization (scaling to unit euclidean length): the *norm* of each vector in the vector space will be normalized to 1. It is necessary for any linear operation of word vectors.

R code:

- Vector: `vec / sqrt(sum(vec^2))`
- Matrix: `mat / sqrt(rowSums(mat^2))`

Usage

```
normalize(x)
```

Arguments

x	A `wordvec` (data.table) or `embed` (matrix), see `data_wordvec_load`.

Value

A `wordvec` (data.table) or `embed` (matrix) with **normalized** word vectors.

Download

Download pre-trained word vectors data (`.RData`): https://psychbruce.github.io/WordVector_RData.pdf

See Also

`as_wordvec` / `as_embed`

`load_wordvec` / `load_embed`

`data_transform`

`data_wordvec_subset`

Examples

```
d = normalize(demodata)
# the same: d = as_wordvec(demodata, normalize=TRUE)
```

`orth_procrustes` *Orthogonal Procrustes rotation for matrix alignment.*

Description

In order to compare word embeddings from different time periods, we must ensure that the embedding matrices are aligned to the same semantic space (coordinate axes). The Orthogonal Procrustes solution (Schönemann, 1966) is commonly used to align historical embeddings over time (Hamilton et al., 2016; Li et al., 2020).

Note that this kind of rotation *does not* change the relative relationships between vectors in the space, and thus *does not* affect semantic similarities or distances within each embedding matrix. But it does influence the semantic relationships between different embedding matrices, and thus would be necessary for some purposes such as the "semantic drift analysis" (e.g., Hamilton et al., 2016; Li et al., 2020).

This function produces the same results as by `cds::orthprocr()`, `psych::Procrustes()`, and `pracma::procrustes()`.

Usage

```
orth_procrustes(M, X)
```

Arguments

`M`, `X`	Two embedding matrices of the same size (rows and columns), can be `embed` or `wordvec` objects. • `M` is the reference (anchor/baseline/target) matrix, e.g., the embedding matrix learned at the later year ($t + 1$). • `X` is the matrix to be transformed/rotated. *Note*: The function automatically extracts only the intersection (overlapped part) of words in `M` and `X` and sorts them in the same order (according to `M`).

Value

A `matrix` or `wordvec` object of `X` after rotation, depending on the class of `M` and `X`.

References

Hamilton, W. L., Leskovec, J., & Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Vol. 1, pp. 1489–1501). Association for Computational Linguistics.

Li, Y., Hills, T., & Hertwig, R. (2020). A brief history of risk. *Cognition, 203*, 104344.

Schönemann, P. H. (1966). A generalized solution of the orthogonal Procrustes problem. *Psychometrika, 31*(1), 1–10.

See Also

`as_wordvec` / `as_embed`

Examples

```
M = matrix(c(0,0,  1,2,  2,0,  3,2,  4,0), ncol=2, byrow=TRUE)
X = matrix(c(0,0, -2,1,  0,2, -2,3,  0,4), ncol=2, byrow=TRUE)
rownames(M) = rownames(X) = cc("A, B, C, D, E")  # words
colnames(M) = colnames(X) = cc("dim1, dim2")  # dimensions

ggplot() +
  geom_path(data=as.data.frame(M), aes(x=dim1, y=dim2),
            color="red") +
  geom_path(data=as.data.frame(X), aes(x=dim1, y=dim2),
            color="blue") +
  coord_equal()

# Usage 1: input two matrices (can be `embed` objects)
XR = orth_procrustes(M, X)
XR  # aligned with M

ggplot() +
  geom_path(data=as.data.frame(XR), aes(x=dim1, y=dim2)) +
  coord_equal()

# Usage 2: input two `wordvec` objects
M.wv = as_wordvec(M)
X.wv = as_wordvec(X)
XR.wv = orth_procrustes(M.wv, X.wv)
XR.wv  # aligned with M.wv

# M and X must have the same set and order of words
# and the same number of word vector dimensions.
# The function extracts only the intersection of words
# and sorts them in the same order according to M.

Y = rbind(X, X[rev(rownames(X)),])
rownames(Y)[1:5] = cc("F, G, H, I, J")
M.wv = as_wordvec(M)
Y.wv = as_wordvec(Y)
M.wv  # words: A, B, C, D, E
Y.wv  # words: F, G, H, I, J, E, D, C, B, A
```

```
YR.wv = orth_procrustes(M.wv, Y.wv)
YR.wv  # aligned with M.wv, with the same order of words
```

`pair_similarity` *Compute a matrix of cosine similarity/distance of word pairs.*

Description

Compute a matrix of cosine similarity/distance of word pairs.

Usage

```
pair_similarity(
  data,
  words = NULL,
  pattern = NULL,
  words1 = NULL,
  words2 = NULL,
  distance = FALSE
)
```

Arguments

`data`	A `wordvec` (data.table) or `embed` (matrix), see `data_wordvec_load`.
`words`	[Option 1] Character string(s).
`pattern`	[Option 2] Regular expression (see `str_subset`). If neither `words` nor `pattern` are specified (i.e., both are `NULL`), then all words in the data will be extracted.
`words1, words2`	[Option 3] Two sets of words for only n1 * n2 word pairs. See examples.
`distance`	Compute cosine distance instead? Defaults to `FALSE` (cosine similarity).

Value

A matrix of pairwise cosine similarity/distance.

Download

Download pre-trained word vectors data (`.RData`): https://psychbruce.github.io/WordVector_RData.pdf

See Also

`cosine_similarity`

`plot_similarity`

`tab_similarity`

`most_similar`

Examples

```
pair_similarity(demodata, c("China", "Chinese"))

pair_similarity(demodata, pattern="^Chi")

pair_similarity(demodata,
                words1=c("China", "Chinese"),
                words2=c("Japan", "Japanese"))
```

---

plot_network *Visualize a (partial correlation) network graph of words.*

Description

Visualize a (partial correlation) network graph of words.

Usage

```
plot_network(
  data,
  words = NULL,
  pattern = NULL,
  index = c("pcor", "cor", "glasso", "sim"),
  alpha = 0.05,
  bonf = FALSE,
  max = NULL,
  node.size = "auto",
  node.group = NULL,
  node.color = NULL,
  label.text = NULL,
  label.size = 1.2,
  label.size.equal = TRUE,
  label.color = "black",
  edge.color = c("#009900", "#BF0000"),
  edge.label = FALSE,
  edge.label.size = 1,
  edge.label.color = NULL,
  edge.label.bg = "white",
  file = NULL,
  width = 10,
  height = 6,
  dpi = 500,
  ...
)
```

Arguments

data	A wordvec (data.table) or embed (matrix), see data_wordvec_load.
words	[Option 1] Character string(s).
pattern	[Option 2] Regular expression (see str_subset). If neither words nor pattern are specified (i.e., both are NULL), then all words in the data will be extracted.
index	Use which index to perform network analysis? Can be "pcor" (partial correlation, default and suggested), "cor" (raw correlation), "glasso" (graphical lasso-estimation of partial correlation matrix using the glasso package), or "sim" (pairwise cosine similarity).
alpha	Significance level to be used for not showing edges. Defaults to 0.05.
bonf	Bonferroni correction of *p* value. Defaults to FALSE.
max	Maximum value for scaling edge widths and colors. Defaults to the highest value of the index. Can be 1 if you want to compare several graphs.
node.size	Node size. Defaults to 8*exp(-nNodes/80)+1.
node.group	Node group(s). Can be a named list (see examples) in which each element is a vector of integers identifying the numbers of the nodes that belong together, or a factor.
node.color	Node color(s). Can be a character vector of colors corresponding to node.group. Defaults to white (if node.group is not specified) or the palette of ggplot2 (if node.group is specified).
label.text	Node label of text. Defaults to original words.
label.size	Node label font size. Defaults to 1.2.
label.size.equal	Make the font size of all labels equal. Defaults to TRUE.
label.color	Node label color. Defaults to "black".
edge.color	Edge colors for positive and negative values, respectively. Defaults to c("#009900", "#BF0000").
edge.label	Edge label of values. Defaults to FALSE.
edge.label.size	Edge label font size. Defaults to 1.
edge.label.color	Edge label color. Defaults to edge.color.
edge.label.bg	Edge label background color. Defaults to "white".
file	File name to be saved, should be png or pdf.
width, height	Width and height (in inches) for the saved file. Defaults to 10 and 6.
dpi	Dots per inch. Defaults to 500 (i.e., file resolution: 4000 * 3000).
...	Other parameters passed to qgraph.

Value

Invisibly return a qgraph object, which further can be plotted using plot().

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

plot_similarity

plot_wordvec_tSNE

Examples

```
d = as_embed(demodata, normalize=TRUE)

words = cc("
man, woman,
he, she,
boy, girl,
father, mother,
mom, dad,
China, Japan
")

plot_network(d, words)

p = plot_network(
  d, words,
  node.group=list(Gender=1:6, Family=7:10, Country=11:12),
  node.color=c("antiquewhite", "lightsalmon", "lightblue"),
  file="network.png")
plot(p)

unlink("network.png")  # delete file for code check

# network analysis with centrality plot (see `qgraph` package)
qgraph::centralityPlot(p, include="all", scale="raw",
                       orderBy="Strength")

# graphical lasso-estimation of partial correlation matrix
plot_network(
  d, words,
  index="glasso",
  # threshold=TRUE,
  node.group=list(Gender=1:6, Family=7:10, Country=11:12),
  node.color=c("antiquewhite", "lightsalmon", "lightblue"))
```

plot_similarity *Visualize cosine similarity of word pairs.*

Description

Visualize cosine similarity of word pairs.

Usage

```
plot_similarity(
  data,
  words = NULL,
  pattern = NULL,
  words1 = NULL,
  words2 = NULL,
  label = "auto",
  value.color = NULL,
  value.percent = FALSE,
  order = c("original", "AOE", "FPC", "hclust", "alphabet"),
  hclust.method = c("complete", "ward", "ward.D", "ward.D2", "single", "average",
    "mcquitty", "median", "centroid"),
  hclust.n = NULL,
  hclust.color = "black",
  hclust.line = 2,
  file = NULL,
  width = 10,
  height = 6,
  dpi = 500,
  ...
)
```

Arguments

data	A wordvec (data.table) or embed (matrix), see data_wordvec_load.
words	[Option 1] Character string(s).
pattern	[Option 2] Regular expression (see str_subset). If neither words nor pattern are specified (i.e., both are NULL), then all words in the data will be extracted.
words1, words2	[Option 3] Two sets of words for only n1 * n2 word pairs. See examples.
label	Position of text labels. Defaults to "auto" (add labels if less than 20 words). Can be TRUE (left and top), FALSE (add no labels of words), or a character string (see the usage of tl.pos in corrplot.
value.color	Color of values added on the plot. Defaults to NULL (add no values).
value.percent	Whether to transform values into percentage style for space saving. Defaults to FALSE.
order	Character, the ordering method of the correlation matrix.

- 'original' for original order (default).
- 'AOE' for the angular order of the eigenvectors.
- 'FPC' for the first principal component order.
- 'hclust' for the hierarchical clustering order.
- 'alphabet' for alphabetical order.

See function corrMatOrder for details.

hclust.method	Character, the agglomeration method to be used when order is hclust. This should be one of 'ward', 'ward.D', 'ward.D2', 'single', 'complete', 'average', 'mcquitty', 'median' or 'centroid'.
hclust.n	Number of rectangles to be drawn on the plot according to the hierarchical clusters, only valid when order="hclust". Defaults to NULL (add no rectangles).
hclust.color	Color of rectangle border, only valid when hclust.n >= 1. Defaults to "black".
hclust.line	Line width of rectangle border, only valid when hclust.n >= 1. Defaults to 2.
file	File name to be saved, should be png or pdf.
width, height	Width and height (in inches) for the saved file. Defaults to 10 and 6.
dpi	Dots per inch. Defaults to 500 (i.e., file resolution: 4000 * 3000).
...	Other parameters passed to corrplot.

Value

Invisibly return a matrix of cosine similarity between each pair of words.

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

cosine_similarity

pair_similarity

tab_similarity

most_similar

plot_network

Examples

```
w1 = cc("king, queen, man, woman")
plot_similarity(demodata, w1)
plot_similarity(demodata, w1,
                value.color="grey",
                value.percent=TRUE)
plot_similarity(demodata, w1,
                value.color="grey",
                order="hclust",
```

```
                hclust.n=2)

plot_similarity(
  demodata,
  words1=cc("man, woman, king, queen"),
  words2=cc("he, she, boy, girl, father, mother"),
  value.color="grey20"
)

w2 = cc("China, Chinese,
         Japan, Japanese,
         Korea, Korean,
         man, woman, boy, girl,
         good, bad, positive, negative")
plot_similarity(demodata, w2,
                order="hclust",
                hclust.n=3)
plot_similarity(demodata, w2,
                order="hclust",
                hclust.n=7,
                file="plot.png")

unlink("plot.png")  # delete file for code check
```

`plot_wordvec` *Visualize word vectors.*

Description

Visualize word vectors.

Usage

```
plot_wordvec(x, dims = NULL, step = 0.05, border = "white")
```

Arguments

`x`	Can be: • a `data.table` returned by `get_wordvec` • a `wordvec` (data.table) or `embed` (matrix) loaded by `data_wordvec_load`
`dims`	Dimensions to be plotted (e.g., `1:100`). Defaults to `NULL` (plot all dimensions).
`step`	Step for value breaks. Defaults to `0.05`.
`border`	Color of tile border. Defaults to `"white"`. To remove the border color, set `border=NA`.

Value

A `ggplot` object.

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

get_wordvec

plot_similarity

plot_wordvec_tSNE

Examples

```
d = as_embed(demodata, normalize=TRUE)

plot_wordvec(d[1:10])

dt = get_wordvec(d, cc("king, queen, man, woman"))
dt[, QUEEN := king - man + woman]
dt[, QUEEN := QUEEN / sqrt(sum(QUEEN^2))]  # normalize
names(dt)[5] = "king - man + woman"
plot_wordvec(dt[, c(1,3,4,5,2)], dims=1:50)

dt = get_wordvec(d, cc("boy, girl, he, she"))
dt[, GIRL := boy - he + she]
dt[, GIRL := GIRL / sqrt(sum(GIRL^2))]  # normalize
names(dt)[5] = "boy - he + she"
plot_wordvec(dt[, c(1,3,4,5,2)], dims=1:50)

dt = get_wordvec(d, cc("
  male, man, boy, he, his,
  female, woman, girl, she, her"))

p = plot_wordvec(dt, dims=1:100)

# if you want to change something:
p + theme(legend.key.height=unit(0.1, "npc"))

# or to save the plot:
ggsave(p, filename="wordvecs.png",
       width=8, height=5, dpi=500)
unlink("wordvecs.png")  # delete file for code check
```

---

plot_wordvec_tSNE *Visualize word vectors with dimensionality reduced using t-SNE.*

Description

Visualize word vectors with dimensionality reduced using the t-Distributed Stochastic Neighbor Embedding (t-SNE) method (i.e., projecting high-dimensional vectors into a low-dimensional vector space), implemented by `Rtsne::Rtsne()`. You should specify a random seed if you expect reproducible results.

Usage

```
plot_wordvec_tSNE(
  x,
  dims = 2,
  perplexity,
  theta = 0.5,
  colors = NULL,
  seed = NULL,
  custom.Rtsne = NULL
)
```

Arguments

`x`	Can be: • a `data.table` returned by `get_wordvec` • a `wordvec` (data.table) or `embed` (matrix) loaded by `data_wordvec_load`
`dims`	Output dimensionality: `2` (default, the most common choice) or `3`.
`perplexity`	Perplexity parameter, should not be larger than (number of words - 1) / 3. Defaults to `floor((length(dt)-1)/3)` (where columns of `dt` are words). See the `Rtsne` package for details.
`theta`	Speed/accuracy trade-off (increase for less accuracy), set to 0 for exact t-SNE. Defaults to 0.5.
`colors`	A character vector specifying (1) the categories of words (for 2-D plot only) or (2) the exact colors of words (for 2-D and 3-D plot). See examples for its usage.
`seed`	Random seed for reproducible results. Defaults to `NULL`.
`custom.Rtsne`	User-defined `Rtsne` object using the same `dt`.

Value

2-D: A `ggplot` object. You may extract the data from this object using `$data`.

3-D: Nothing but only the data was invisibly returned, because `rgl::plot3d()` is "called for the side effect of drawing the plot" and thus cannot return any 3-D plot object.

Download

Download pre-trained word vectors data (`.RData`): https://psychbruce.github.io/WordVector_RData.pdf

References

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504–507.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*, 2579–2605.

See Also

plot_wordvec

plot_network

Examples

```
d = as_embed(demodata, normalize=TRUE)

dt = get_wordvec(d, cc("
  man, woman,
  king, queen,
  China, Beijing,
  Japan, Tokyo"))

## 2-D (default):
plot_wordvec_tSNE(dt, seed=1234)

plot_wordvec_tSNE(dt, seed=1234)$data

colors = c(rep("#2B579A", 4), rep("#B7472A", 4))
plot_wordvec_tSNE(dt, colors=colors, seed=1234)

category = c(rep("gender", 4), rep("country", 4))
plot_wordvec_tSNE(dt, colors=category, seed=1234) +
  scale_x_continuous(limits=c(-200, 200),
                     labels=function(x) x/100) +
  scale_y_continuous(limits=c(-200, 200),
                     labels=function(x) x/100) +
  scale_color_manual(values=c("#B7472A", "#2B579A"))

## 3-D:
colors = c(rep("#2B579A", 4), rep("#B7472A", 4))
plot_wordvec_tSNE(dt, dims=3, colors=colors, seed=1)
```

---

sum_wordvec *Calculate the sum vector of multiple words.*

Description

Calculate the sum vector of multiple words.

Usage

```
sum_wordvec(data, x = NULL, verbose = TRUE)
```

Arguments

data	A wordvec (data.table) or embed (matrix), see data_wordvec_load.
x	Can be: • NULL: use the sum of all word vectors in data • a single word: "China" • a list of words: c("king", "queen") cc(" king , queen ; man | woman") • an R formula (~ xxx) specifying words that positively and negatively contribute to the similarity (for word analogy): ~ boy - he + she ~ king - man + woman ~ Beijing - China + Japan
verbose	Print information to the console? Defaults to TRUE.

Value

Normalized sum vector.

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

normalize

most_similar

dict_expand

dict_reliability

Examples

```
sum_wordvec(normalize(demodata), ~ king - man + woman)
```

tab_similarity *Tabulate cosine similarity/distance of word pairs.*

Description

Tabulate cosine similarity/distance of word pairs.

Usage

```
tab_similarity(
  data,
  words = NULL,
  pattern = NULL,
  words1 = NULL,
  words2 = NULL,
  unique = FALSE,
  distance = FALSE
)
```

Arguments

data	A wordvec (data.table) or embed (matrix), see data_wordvec_load.
words	[Option 1] Character string(s).
pattern	[Option 2] Regular expression (see str_subset). If neither words nor pattern are specified (i.e., both are NULL), then all words in the data will be extracted.
words1, words2	[Option 3] Two sets of words for only n1 * n2 word pairs. See examples.
unique	Return unique word pairs (TRUE) or all pairs with duplicates (FALSE; default).
distance	Compute cosine distance instead? Defaults to FALSE (cosine similarity).

Value

A data.table of words, word pairs, and their cosine similarity (cos_sim) or cosine distance (cos_dist).

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

See Also

cosine_similarity

pair_similarity

plot_similarity

most_similar

test_WEAT

test_RND

Examples

```
tab_similarity(demodata, cc("king, queen, man, woman"))
tab_similarity(demodata, cc("king, queen, man, woman"),
               unique=TRUE)

tab_similarity(demodata, cc("Beijing, China, Tokyo, Japan"))
tab_similarity(demodata, cc("Beijing, China, Tokyo, Japan"),
               unique=TRUE)

## only n1 * n2 word pairs across two sets of words
tab_similarity(demodata,
               words1=cc("king, queen, King, Queen"),
               words2=cc("man, woman"))
```

---

test_RND *Relative Norm Distance (RND) analysis.*

Description

Tabulate data and conduct the permutation test of significance for the *Relative Norm Distance* (RND; also known as *Relative Euclidean Distance*). This is an alternative method to Single-Category WEAT.

Usage

```
test_RND(
  data,
  T1,
  A1,
  A2,
  use.pattern = FALSE,
  labels = list(),
  p.perm = TRUE,
  p.nsim = 10000,
  p.side = 2,
  seed = NULL
)
```

Arguments

data	A wordvec (data.table) or embed (matrix), see data_wordvec_load.
T1	Target words of a single category (a vector of words or a pattern of regular expression).

A1, A2	Attribute words (a vector of words or a pattern of regular expression). Both must be specified.
use.pattern	Defaults to FALSE (using a vector of words). If you use regular expression in T1, T2, A1, and A2, please specify this argument as TRUE.
labels	Labels for target and attribute concepts (a named list), such as (the default) list(T1="Target", A1="Attrib1", A2="Attrib2").
p.perm	Permutation test to get exact or approximate p value of the overall effect. Defaults to TRUE. See also the sweater package.
p.nsim	Number of samples for resampling in permutation test. Defaults to 10000. If p.nsim is larger than the number of all possible permutations (rearrangements of data), then it will be ignored and an exact permutation test will be conducted. Otherwise (in most cases for real data and always for SC-WEAT), a resampling test is performed, which takes much less computation time and produces the approximate p value (comparable to the exact one).
p.side	One-sided (1) or two-sided (2) p value. Defaults to 2. In Caliskan et al.'s (2017) article, they reported one-sided p value for WEAT. Here, I suggest reporting two-sided p value as a more conservative estimate. The users take the full responsibility for the choice. • The one-sided p value is calculated as the proportion of sampled permutations where the difference in means is greater than the test statistic. • The two-sided p value is calculated as the proportion of sampled permutations where the absolute difference is greater than the test statistic.
seed	Random seed for reproducible results of permutation test. Defaults to NULL.

Value

A list object of new class rnd:

words.valid Valid (actually matched) words

words.not.found Words not found

data.raw A data.table of (absolute and relative) norm distances

eff.label Description for the difference between the two attribute concepts

eff.type Effect type: RND

eff Raw effect and p value (if p.perm=TRUE)

eff.interpretation Interpretation of the RND score

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

References

Garg, N., Schiebinger, L., Jurafsky, D., & Zou, J. (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences, 115*(16), E3635–E3644.

Bhatia, N., & Bhatia, S. (2021). Changes in gender stereotypes over time: A computational analysis. *Psychology of Women Quarterly, 45*(1), 106–125.

See Also

tab_similarity

dict_expand

dict_reliability

test_WEAT

Examples

```
rnd = test_RND(
  demodata,
  labels=list(T1="Occupation", A1="Male", A2="Female"),
  T1=cc("
    architect, boss, leader, engineer, CEO, officer, manager,
    lawyer, scientist, doctor, psychologist, investigator,
    consultant, programmer, teacher, clerk, counselor,
    salesperson, therapist, psychotherapist, nurse"),
  A1=cc("male, man, boy, brother, he, him, his, son"),
  A2=cc("female, woman, girl, sister, she, her, hers, daughter"),
  seed=1)
rnd
```

---

test_WEAT *Word Embedding Association Test (WEAT) and Single-Category WEAT.*

Description

Tabulate data (cosine similarity and standardized effect size) and conduct the permutation test of significance for the *Word Embedding Association Test* (WEAT) and *Single-Category Word Embedding Association Test* (SC-WEAT).

- For WEAT, two-samples permutation test is conducted (i.e., rearrangements of data).
- For SC-WEAT, one-sample permutation test is conducted (i.e., rearrangements of +/- signs to data).

Usage

```
test_WEAT(
  data,
  T1,
  T2,
  A1,
  A2,
  use.pattern = FALSE,
  labels = list(),
  p.perm = TRUE,
  p.nsim = 10000,
  p.side = 2,
  seed = NULL,
  pooled.sd = "Caliskan"
)
```

Arguments

`data`	A `wordvec` (data.table) or `embed` (matrix), see `data_wordvec_load`.
`T1`, `T2`	Target words (a vector of words or a pattern of regular expression). If only `T1` is specified, it will tabulate data for single-category WEAT (SC-WEAT).
`A1`, `A2`	Attribute words (a vector of words or a pattern of regular expression). Both must be specified.
`use.pattern`	Defaults to `FALSE` (using a vector of words). If you use regular expression in `T1`, `T2`, `A1`, and `A2`, please specify this argument as `TRUE`.
`labels`	Labels for target and attribute concepts (a named `list`), such as (the default) `list(T1="Target1", T2="Target2", A1="Attrib1", A2="Attrib2")`.
`p.perm`	Permutation test to get exact or approximate *p* value of the overall effect. Defaults to `TRUE`. See also the `sweater` package.
`p.nsim`	Number of samples for resampling in permutation test. Defaults to `10000`. If `p.nsim` is larger than the number of all possible permutations (rearrangements of data), then it will be ignored and an exact permutation test will be conducted. Otherwise (in most cases for real data and always for SC-WEAT), a resampling test is performed, which takes much less computation time and produces the approximate *p* value (comparable to the exact one).
`p.side`	One-sided (`1`) or two-sided (`2`) *p* value. Defaults to `2`. In Caliskan et al.'s (2017) article, they reported one-sided *p* value for WEAT. Here, I suggest reporting two-sided *p* value as a more conservative estimate. The users take the full responsibility for the choice. • The one-sided *p* value is calculated as the proportion of sampled permutations where the difference in means is greater than the test statistic. • The two-sided *p* value is calculated as the proportion of sampled permutations where the absolute difference is greater than the test statistic.
`seed`	Random seed for reproducible results of permutation test. Defaults to `NULL`.
`pooled.sd`	Method used to calculate the pooled *SD* for effect size estimate in WEAT.

- Defaults to "Caliskan": sd(data.diff$cos_sim_diff), which is highly suggested and identical to Caliskan et al.'s (2017) original approach.
- Otherwise specified, it will calculate the pooled *SD* as: $\sqrt{[(n_1-1)*\sigma_1^2+(n_2-1)*\sigma_2^2]/(n_1+n_2}$ This is **NOT suggested** because it may *overestimate* the effect size, especially when there are only a few T1 and T2 words that have small variances.

Value

A list object of new class weat:

words.valid Valid (actually matched) words

words.not.found Words not found

data.raw A data.table of cosine similarities between all word pairs

data.mean A data.table of *mean* cosine similarities *across* all attribute words

data.diff A data.table of *differential* mean cosine similarities *between* the two attribute concepts

eff.label Description for the difference between the two attribute concepts

eff.type Effect type: WEAT or SC-WEAT

eff Raw effect, standardized effect size, and p value (if p.perm=TRUE)

Download

Download pre-trained word vectors data (.RData): https://psychbruce.github.io/WordVector_RData.pdf

References

Caliskan, A., Bryson, J. J., & Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science, 356*(6334), 183–186.

See Also

tab_similarity

dict_expand

dict_reliability

test_RND

Examples

```
## cc() is more convenient than c()!

weat = test_WEAT(
  demodata,
  labels=list(T1="King", T2="Queen", A1="Male", A2="Female"),
  T1=cc("king, King"),
  T2=cc("queen, Queen"),
  A1=cc("male, man, boy, brother, he, him, his, son"),
```

```
  A2=cc("female, woman, girl, sister, she, her, hers, daughter"),
  seed=1)
weat

sc_weat = test_WEAT(
  demodata,
  labels=list(T1="Occupation", A1="Male", A2="Female"),
  T1=cc("
    architect, boss, leader, engineer, CEO, officer, manager,
    lawyer, scientist, doctor, psychologist, investigator,
    consultant, programmer, teacher, clerk, counselor,
    salesperson, therapist, psychotherapist, nurse"),
  A1=cc("male, man, boy, brother, he, him, his, son"),
  A2=cc("female, woman, girl, sister, she, her, hers, daughter"),
  seed=1)
sc_weat

## Not run:

## the same as the first example, but using regular expression
weat = test_WEAT(
  demodata,
  labels=list(T1="King", T2="Queen", A1="Male", A2="Female"),
  use.pattern=TRUE,  # use regular expression below
  T1="^[kK]ing$",
  T2="^[qQ]ueen$",
  A1="^male$|^man$|^boy$|^brother$|^he$|^him$|^his$|^son$",
  A2="^female$|^woman$|^girl$|^sister$|^she$|^her$|^hers$|^daughter$",
  seed=1)
weat

## replicating Caliskan et al.'s (2017) results
## WEAT7 (Table 1): d = 1.06, p = .018
## (requiring installation of the `sweater` package)
Caliskan.WEAT7 = test_WEAT(
  as_wordvec(sweater::glove_math),
  labels=list(T1="Math", T2="Arts", A1="Male", A2="Female"),
  T1=cc("math, algebra, geometry, calculus, equations, computation, numbers, addition"),
  T2=cc("poetry, art, dance, literature, novel, symphony, drama, sculpture"),
  A1=cc("male, man, boy, brother, he, him, his, son"),
  A2=cc("female, woman, girl, sister, she, her, hers, daughter"),
  p.side=1, seed=1234)
Caliskan.WEAT7
# d = 1.055, p = .0173 (= 173 counts / 10000 permutation samples)

## replicating Caliskan et al.'s (2017) supplemental results
## WEAT7 (Table S1): d = 0.97, p = .027
Caliskan.WEAT7.supp = test_WEAT(
  demodata,
  labels=list(T1="Math", T2="Arts", A1="Male", A2="Female"),
  T1=cc("math, algebra, geometry, calculus, equations, computation, numbers, addition"),
  T2=cc("poetry, art, dance, literature, novel, symphony, drama, sculpture"),
  A1=cc("male, man, boy, brother, he, him, his, son"),
```

```
  A2=cc("female, woman, girl, sister, she, her, hers, daughter"),
  p.side=1, seed=1234)
Caliskan.WEAT7.supp
# d = 0.966, p = .0221 (= 221 counts / 10000 permutation samples)

## End(Not run)
```

`text_init` *Install required Python modules in a new conda environment and initialize the environment, necessary for all* `text_*` *functions designed for contextualized word embeddings.*

Description

Install required Python modules in a new conda environment and initialize the environment, necessary for all `text_*` functions designed for contextualized word embeddings.

Usage

```
text_init()
```

Details

Users may first need to manually install Anaconda or Miniconda.

The R package `text` (`https://www.r-text.org/`) enables users access to HuggingFace Transformers models in R, through the R package `reticulate` as an interface to Python and the Python modules `torch` and `transformers`.

For advanced usage, see

- `text::textrpp_install()`
- `text::textrpp_install_virtualenv()`
- `text::textrpp_uninstall()`
- `text::textrpp_initialize()`

See Also

`text_model_download`

`text_model_remove`

`text_to_vec`

`text_unmask`

Examples

```
## Not run:
text_init()

# You may need to specify the version of Python:
# RStudio -> Tools -> Global/Project Options
# -> Python -> Select -> Conda Environments
# -> Choose ".../textrpp_condaenv/python.exe"

## End(Not run)
```

`text_model_download` *Download pre-trained language models from HuggingFace.*

Description

Download pre-trained language models (Transformers Models, such as GPT, BERT, RoBERTa, DeBERTa, DistilBERT, etc.) from HuggingFace to your local ".cache" folder ("C:/Users/[YourUserName]/.cache/"). The models will never be removed unless you run `text_model_remove`.

Usage

```
text_model_download(model = NULL)
```

Arguments

`model`	Character string(s) specifying the pre-trained language model(s) to be downloaded. For a full list of options, see HuggingFace. Defaults to download nothing and check currently downloaded models. Example choices: • `"gpt2"` (50257 vocab, 768 dims, 12 layers) • `"openai-gpt"` (40478 vocab, 768 dims, 12 layers) • `"bert-base-uncased"` (30522 vocab, 768 dims, 12 layers) • `"bert-large-uncased"` (30522 vocab, 1024 dims, 24 layers) • `"bert-base-cased"` (28996 vocab, 768 dims, 12 layers) • `"bert-large-cased"` (28996 vocab, 1024 dims, 24 layers) • `"bert-base-chinese"` (21128 vocab, 768 dims, 12 layers) • `"bert-base-multilingual-cased"` (119547 vocab, 768 dims, 12 layers) • `"distilbert-base-uncased"` (30522 vocab, 768 dims, 6 layers) • `"distilbert-base-cased"` (28996 vocab, 768 dims, 6 layers) • `"distilbert-base-multilingual-cased"` (119547 vocab, 768 dims, 6 layers) • `"albert-base-v2"` (30000 vocab, 768 dims, 12 layers) • `"albert-large-v2"` (30000 vocab, 1024 dims, 24 layers)

- "roberta-base" (50265 vocab, 768 dims, 12 layers)
- "roberta-large" (50265 vocab, 1024 dims, 24 layers)
- "xlm-roberta-base" (250002 vocab, 768 dims, 12 layers)
- "xlm-roberta-large" (250002 vocab, 1024 dims, 24 layers)
- "xlnet-base-cased" (32000 vocab, 768 dims, 12 layers)
- "xlnet-large-cased" (32000 vocab, 1024 dims, 24 layers)
- "microsoft/deberta-v3-base" (128100 vocab, 768 dims, 12 layers)
- "microsoft/deberta-v3-large" (128100 vocab, 1024 dims, 24 layers)
- ... (see https://huggingface.co/models)

Value

Invisibly return the names of all downloaded models.

See Also

text_init

text_model_remove

text_to_vec

text_unmask

Examples

```
## Not run:
# text_init()  # initialize the environment

text_model_download()  # check downloaded models
text_model_download(c(
  "bert-base-uncased",
  "bert-base-cased",
  "bert-base-multilingual-cased"
))

## End(Not run)
```

---

text_model_remove *Remove downloaded models from the local .cache folder.*

Description

Remove downloaded models from the local .cache folder.

Usage

```
text_model_remove(model = NULL)
```

Arguments

model	Model name. See text_model_download. Defaults to automatically find all downloaded models in the .cache folder.

See Also

text_init

text_model_download

text_to_vec

text_unmask

Examples

```
## Not run:
# text_init()  # initialize the environment

text_model_remove()

## End(Not run)
```

---

text_to_vec *Extract contextualized word embeddings from transformers (pre-trained language models).*

Description

Extract hidden layers from a language model and aggregate them to get token (roughly word) embeddings and text embeddings (all reshaped to embed matrix). It is a wrapper function of text::textEmbed().

Usage

```
text_to_vec(
  text,
  model,
  layers = "all",
  layer.to.token = "concatenate",
  token.to.word = TRUE,
  token.to.text = TRUE,
  encoding = "UTF-8",
  ...
)
```

Arguments

text	Can be: • a character string or vector of text (usually sentences) • a data frame with at least one character variable (for text from all character variables in a given data frame) • a file path on disk containing text
model	Model name at HuggingFace. See text_model_download. If the model has not been downloaded, it would automatically download the model.
layers	Layers to be extracted from the model, which are then aggregated in the function text::textEmbedLayerAggregation(). Defaults to "all" which extracts all layers. You may extract only the layers you need (e.g., 11:12). Note that layer 0 is the *decontextualized* input layer (i.e., not comprising hidden states).
layer.to.token	Method to aggregate hidden layers to each token. Defaults to "concatenate", which links together each word embedding layer to one long row. Options include "mean", "min", "max", and "concatenate".
token.to.word	Aggregate subword token embeddings (if whole word is out of vocabulary) to whole word embeddings. Defaults to TRUE, which sums up subword token embeddings.
token.to.text	Aggregate token embeddings to each text. Defaults to TRUE, which averages all token embeddings. If FALSE, the text embedding will be the token embedding of [CLS] (the special token that is used to represent the beginning of a text sequence).
encoding	Text encoding (only used if text is a file). Defaults to "UTF-8".
...	Other parameters passed to text::textEmbed().

Value

A list of:

token.embed Token (roughly word) embeddings

text.embed Text embeddings, aggregated from token embeddings

See Also

text_init

text_model_download

text_model_remove

text_unmask

Examples

```
## Not run:
# text_init()  # initialize the environment

text = c("Download models from HuggingFace",
```

```
          "Chinese are East Asian",
          "Beijing is the capital of China")
embed = text_to_vec(text, model="bert-base-cased", layers=c(0, 12))
embed

embed1 = embed$token.embed[[1]]
embed2 = embed$token.embed[[2]]
embed3 = embed$token.embed[[3]]

View(embed1)
View(embed2)
View(embed3)
View(embed$text.embed)

plot_similarity(embed1, value.color="grey")
plot_similarity(embed2, value.color="grey")
plot_similarity(embed3, value.color="grey")
plot_similarity(rbind(embed1, embed2, embed3))

## End(Not run)
```

`text_unmask` *<Deprecated> Fill in the blank mask(s) in a query (sentence).*

Description

Note: This function has been deprecated and will not be updated since I have developed new package FMAT as the integrative toolbox of *Fill-Mask Association Test* (FMAT).

Predict the probably correct masked token(s) in a sequence, based on the Python module `transformers`.

Usage

```
text_unmask(query, model, targets = NULL, topn = 5)
```

Arguments

`query`	A query (sentence/prompt) with masked token(s) `[MASK]`. Multiple queries are also supported. See examples.
`model`	Model name at HuggingFace. See `text_model_download`. If the model has not been downloaded, it would automatically download the model.
`targets`	Specific target word(s) to be filled in the blank `[MASK]`. Defaults to `NULL` (i.e., return `topn`). If specified, then `topn` will be ignored (see examples).
`topn`	Number of the most likely predictions to return. Defaults to `5`. If `targets` is specified, then it will automatically change to the length of `targets`.

Details

Masked language modeling is the task of masking some of the words in a sentence and predicting which words should replace those masks. These models are useful when we want to get a statistical understanding of the language in which the model is trained in. See https://huggingface.co/tasks/fill-mask for details.

Value

A `data.table` of query results:

`query_id` **(if there are more than one** `query`**)** `query` ID (indicating multiple queries)

`mask_id` **(if there are more than one** `[MASK]` **in** `query`**)** `[MASK]` ID (position in sequence, indicating multiple masks)

`prob` Probability of the predicted token in the sequence

`token_id` Predicted token ID (to replace `[MASK]`)

`token` Predicted token (to replace `[MASK]`)

`sequence` Complete sentence with the predicted token

See Also

`text_init`

`text_model_download`

`text_model_remove`

`text_to_vec`

Examples

```
## Not run:
# text_init()  # initialize the environment

model = "distilbert-base-cased"

text_unmask("Beijing is the [MASK] of China.", model)

# multiple [MASK]s:
text_unmask("Beijing is the [MASK] [MASK] of China.", model)

# multiple queries:
text_unmask(c("The man worked as a [MASK].",
              "The woman worked as a [MASK]."),
            model)

# specific targets:
text_unmask("The [MASK] worked as a nurse.", model,
            targets=c("man", "woman"))

## End(Not run)
```

tokenize *Tokenize raw text for training word embeddings.*

Description

Tokenize raw text for training word embeddings.

Usage

```
tokenize(
  text,
  tokenizer = text2vec::word_tokenizer,
  split = " ",
  remove = "_|'|<br/>|<br />|e\\.g\\.|i\\.e\\.",
  encoding = "UTF-8",
  simplify = TRUE,
  verbose = TRUE
)
```

Arguments

text	A character vector of text, or a file path on disk containing text.
tokenizer	Function used to tokenize the text. Defaults to text2vec::word_tokenizer.
split	Separator between tokens, only used when simplify=TRUE. Defaults to " ".
remove	Strings (in regular expression) to be removed from the text. Defaults to "_|'| | |e\\.g\\.|i\\.e\\.". You may turn off this by specifying remove=NULL.
encoding	Text encoding (only used if text is a file). Defaults to "UTF-8".
simplify	Return a character vector (TRUE) or a list of character vectors (FALSE). Defaults to TRUE.
verbose	Print information to the console? Defaults to TRUE.

Value

- simplify=TRUE: A tokenized character vector, with each element as a sentence.
- simplify=FALSE: A list of tokenized character vectors, with each element as a vector of tokens in a sentence.

See Also

train_wordvec

Examples

```
txt1 = c(
  "I love natural language processing (NLP)!",
  "I've been in this city for 10 years. I really like here!",
  "However, my computer is not among the \"Top 10\" list."
)
tokenize(txt1, simplify=FALSE)
tokenize(txt1) %>% cat(sep="\n----\n")

txt2 = text2vec::movie_review$review[1:5]
texts = tokenize(txt2)

txt2[1]
texts[1:20]  # all sentences in txt2[1]
```

`train_wordvec` *Train static word embeddings using the Word2Vec, GloVe, or FastText algorithm.*

Description

Train static word embeddings using the Word2Vec, GloVe, or FastText algorithm with multi-threading.

Usage

```
train_wordvec(
  text,
  method = c("word2vec", "glove", "fasttext"),
  dims = 300,
  window = 5,
  min.freq = 5,
  threads = 8,
  model = c("skip-gram", "cbow"),
  loss = c("ns", "hs"),
  negative = 5,
  subsample = 1e-04,
  learning = 0.05,
  ngrams = c(3, 6),
  x.max = 10,
  convergence = -1,
  stopwords = character(0),
  encoding = "UTF-8",
  tolower = FALSE,
  normalize = FALSE,
  iteration,
  tokenizer,
```

```
  remove,
  file.save,
  compress = "bzip2",
  verbose = TRUE
)
```

Arguments

`text`	A character vector of text, or a file path on disk containing text.
`method`	Training algorithm: • `"word2vec"` (default): using the `word2vec` package • `"glove"`: using the `rsparse` and `text2vec` packages • `"fasttext"`: using the `fastTextR` package
`dims`	Number of dimensions of word vectors to be trained. Common choices include 50, 100, 200, 300, and 500. Defaults to `300`.
`window`	Window size (number of nearby words behind/ahead the current word). It defines how many surrounding words to be included in training: [window] words behind and [window] words ahead ([window]*2 in total). Defaults to `5`.
`min.freq`	Minimum frequency of words to be included in training. Words that appear less than this value of times will be excluded from vocabulary. Defaults to `5` (take words that appear at least five times).
`threads`	Number of CPU threads used for training. A modest value produces the fastest training. Too many threads are not always helpful. Defaults to `8`.
`model`	**<Only for Word2Vec / FastText>** Learning model architecture: • `"skip-gram"` (default): Skip-Gram, which predicts surrounding words given the current word • `"cbow"`: Continuous Bag-of-Words, which predicts the current word based on the context
`loss`	**<Only for Word2Vec / FastText>** Loss function (computationally efficient approximation): • `"ns"` (default): Negative Sampling • `"hs"`: Hierarchical Softmax
`negative`	**<Only for Negative Sampling in Word2Vec / FastText>** Number of negative examples. Values in the range 5~20 are useful for small training datasets, while for large datasets the value can be as small as 2~5. Defaults to `5`.
`subsample`	**<Only for Word2Vec / FastText>** Subsampling of frequent words (threshold for occurrence of words). Those that appear with higher frequency in the training data will be randomly downsampled. Defaults to `0.0001` (`1e-04`).
`learning`	**<Only for Word2Vec / FastText>** Initial (starting) learning rate, also known as alpha. Defaults to `0.05`.

ngrams	**<Only for FastText>** Minimal and maximal ngram length. Defaults to `c(3, 6)`.
x.max	**<Only for GloVe>** Maximum number of co-occurrences to use in the weighting function. Defaults to `10`.
convergence	**<Only for GloVe>** Convergence tolerance for SGD iterations. Defaults to `-1`.
stopwords	**<Only for Word2Vec / GloVe>** A character vector of stopwords to be excluded from training.
encoding	Text encoding. Defaults to `"UTF-8"`.
tolower	Convert all upper-case characters to lower-case? Defaults to `FALSE`.
normalize	Normalize all word vectors to unit length? Defaults to `FALSE`. See `normalize`.
iteration	Number of training iterations. More iterations makes a more precise model, but computational cost is linearly proportional to iterations. Defaults to `5` for Word2Vec and FastText while `10` for GloVe.
tokenizer	Function used to tokenize the text. Defaults to `text2vec::word_tokenizer`.
remove	Strings (in regular expression) to be removed from the text. Defaults to `"_|'| | |e\\.g\\.|i\\.e\\."`. You may turn off this by specifying `remove=NULL`.
file.save	File name of to-be-saved R data (must be .RData).
compress	Compression method for the saved file. Defaults to `"bzip2"`. Options include: • `1` or `"gzip"`: modest file size (fastest) • `2` or `"bzip2"`: small file size (fast) • `3` or `"xz"`: minimized file size (slow)
verbose	Print information to the console? Defaults to `TRUE`.

Value

A `wordvec` (data.table) with three variables: `word`, `vec`, `freq`.

Download

Download pre-trained word vectors data (`.RData`): https://psychbruce.github.io/WordVector_RData.pdf

References

All-in-one package:

- https://CRAN.R-project.org/package=wordsalad

Word2Vec:

- https://code.google.com/archive/p/word2vec/
- https://CRAN.R-project.org/package=word2vec

- https://github.com/maxoodf/word2vec

GloVe:

- https://nlp.stanford.edu/projects/glove/
- https://text2vec.org/glove.html
- https://CRAN.R-project.org/package=text2vec
- https://CRAN.R-project.org/package=rsparse

FastText:

- https://fasttext.cc/
- https://CRAN.R-project.org/package=fastTextR

See Also

tokenize

Examples

```
review = text2vec::movie_review  # a data.frame'
text = review$review

## Note: All the examples train 50 dims for faster code check.

## Word2Vec (SGNS)
dt1 = train_wordvec(
  text,
  method="word2vec",
  model="skip-gram",
  dims=50, window=5,
  normalize=TRUE)

dt1
most_similar(dt1, "Ive")  # evaluate performance
most_similar(dt1, ~ man - he + she, topn=5)  # evaluate performance
most_similar(dt1, ~ boy - he + she, topn=5)  # evaluate performance

## GloVe
dt2 = train_wordvec(
  text,
  method="glove",
  dims=50, window=5,
  normalize=TRUE)

dt2
most_similar(dt2, "Ive")  # evaluate performance
most_similar(dt2, ~ man - he + she, topn=5)  # evaluate performance
most_similar(dt2, ~ boy - he + she, topn=5)  # evaluate performance

## FastText
dt3 = train_wordvec(
```

```
  text,
  method="fasttext",
  model="skip-gram",
  dims=50, window=5,
  normalize=TRUE)

dt3
most_similar(dt3, "Ive")  # evaluate performance
most_similar(dt3, ~ man - he + she, topn=5)  # evaluate performance
most_similar(dt3, ~ boy - he + she, topn=5)  # evaluate performance
```

Index