Package 'PsychWordVec'

August 22, 2022

Title Word Embedding Research Framework for Psychological Science

Version 0.1.0

Date 2022-08-21

Author Han-Wu-Shuang Bao [aut, cre]

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

Description An integrated toolkit of word embedding research that provides:
(1) a collection of 'pre-trained' word vectors in the '.RData'
compressed format <https://psychbruce.github.io/WordVector_RData.pdf>;
(2) a variety 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>;
(4) a set of training methods to locally train word vectors from
text corpora, including 'Word2Vec' <arXiv:1301.3781>,
'GloVe' <doi:10.3115/v1/D14-1162>, and 'FastText' <arXiv:1607.04606>.

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, vroom, ggplot2, ggrepel,
Rtsne, rgl, rsparse, text2vec, word2vec, fastTextR

Suggests wordsalad, sweater

RoxygenNote 7.2.1

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-22 15:20:02 UTC

R topics documented:

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

Description

Cosine similarity/distance between two vectors.

Usage

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

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

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

cosine_similarity(v1=c(1,1,1), v2=c(2,2,2), distance=TRUE)  # 0
cosine_similarity(v1=c(1,4,1), v2=c(4,1,1), distance=TRUE)  # 0.5
cosine_similarity(v1=c(1,1,0), v2=c(0,0,1), distance=TRUE)  # 1
```

---

data_transform *Transform plain text data of word vectors into a compressed ".RData" file.*

Description

Transform plain text data of word vectors into 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,
  sep = " ",
  header = "auto",
  encoding = "auto",
  compress = "bzip2",
  compress.level = 9,
  verbose = TRUE
)
```

Arguments

file.load	File name of raw data (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).
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 data.table (of new class wordvec) with two variables: word and vec.

Download

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

See Also

data_wordvec_load

data_wordvec_normalize

data_wordvec_reshape

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 from an ".RData" file.*

Description

Load word vectors data from an ".RData" file.

Usage

```
data_wordvec_load(file.load, normalize = FALSE, verbose = TRUE)
```

Arguments

`file.load`	File name (must be .RData transformed by `data_transform`).
`normalize`	Normalize all word vectors to unit length? Defaults to `FALSE`. See `data_wordvec_normalize`.
`verbose`	Print information to the console? Defaults to `TRUE`.

Value

A `data.table` (of new class `wordvec`) with two variables:

`word` words (tokens)

`vec` **raw** *or* **normalized** word vectors

Download

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

See Also

`data_transform`

`data_wordvec_normalize`

`data_wordvec_reshape`

`data_wordvec_subset`

Examples

```
## 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 = data_wordvec_load("GloVe/glove_wiki_50d.RData")

## End(Not run)
```

`data_wordvec_normalize`

Normalize all word vectors to unit length.

Description

L2-normalization (scaling to unit euclidean length): the *norm* of each vector in the vector space will be normalized to 1.

R formula: `normalized_vec = vec / sqrt(sum(vec^2))`

Note: Normalization does not change the results of cosine similarity and can make the computation faster.

Usage

```
data_wordvec_normalize(data, verbose = TRUE)
```

Arguments

`data`	A `data.table` (of new class `wordvec`) loaded by `data_wordvec_load`.
`verbose`	Print information to the console? Defaults to `TRUE`.

Value

A `data.table` (of new class `wordvec`) with **normalized** word vectors.

Download

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

See Also

data_transform

data_wordvec_load

data_wordvec_reshape

data_wordvec_subset

Examples

```
d = data_wordvec_normalize(demodata)

data_wordvec_normalize(d)  # already normalized
```

---

data_wordvec_reshape *Reshape word vectors data.*

Description

Reshape word vectors data from dense (`data.table`) to plain (`matrix`) or vice versa.

Usage

```
data_wordvec_reshape(
  data,
  to = c("plain", "dense"),
  normalize = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	Data to be reshaped. See examples.
`to`	Options include: • `"plain"` (default) reshapes the data from `data.table` (with two variables `word` and `vec`, loaded by data_wordvec_load) to `matrix` (with dimensions as columns and words as row names). • `"dense"` just does the reverse.
`normalize`	Normalize all word vectors to unit length? Defaults to `FALSE`. See data_wordvec_normalize.
`verbose`	Print information to the console? Defaults to `TRUE`.

Value

A `data.table` (dense) or `matrix` (plain) of word vectors.

Download

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

See Also

data_transform

data_wordvec_load

data_wordvec_normalize

data_wordvec_subset

Examples

```
d = head(demodata, 10)
d

d.plain = data_wordvec_reshape(d, to="plain")
d.plain

d.dense = data_wordvec_reshape(d.plain, to="dense")
d.dense  # identical to `d`
```

---

data_wordvec_subset *Extract a subset of word vectors data.*

Description

Extract a subset of word vectors data. You may specify either a data.table loaded by data_wordvec_load) or an .RData file transformed by data_transform).

Usage

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

Arguments

x	Can be one of the following: • a data.table (of new class wordvec) loaded by data_wordvec_load • an .RData file transformed by data_transform
words	[Option 1] Word strings (NULL; a single word; a vector of words).
pattern	[Option 2] Pattern of regular expression (see str_subset).
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.

Value

A subset of word vectors data of valid (available) words.

Download

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

See Also

data_transform

data_wordvec_load

data_wordvec_normalize

data_wordvec_reshape

Examples

```
## specify `x` as a data.table:
d = data_wordvec_subset(demodata, c("China", "Japan", "Korea"))
d

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

## load the subset:
d.subset = data_wordvec_load("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 = data_wordvec_load("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 (corpus: Google News; algorithm: word2vec; vocabulary: 8000; dimensions: 300).*

Description

This demo data contains a sample of 8000 English words with their 300-d word embeddings (word vectors) 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)
head(demodata, 10)
data_wordvec_normalize(demodata)
```

get_wordvec *Extract the word vector of a single word.*

Description

Extract the word vector of a single word.

Usage

```
get_wordvec(data, word)
```

Arguments

data	A data.table (of new class wordvec) loaded by data_wordvec_load.
word	Word string (a single word).

Value

A numeric vector of the word (or NA if the word does not appear in the data).

Download

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

See Also

get_wordvecs

plot_wordvec

plot_wordvec_tSNE

Examples

```
d = data_wordvec_normalize(demodata)

v1 = get_wordvec(demodata, "China")  # raw vector
v2 = get_wordvec(d, "China")  # normalized vector
cor(v1, v2)
cosine_similarity(v1, v2)
```

get_wordvecs *Extract the word vectors of multiple words.*

Description

Extract the word vectors of multiple words, using either wordlist (a vector of words; using `words`) or regular expression (a pattern of words; using `pattern`). If both the `words` and `pattern` arguments are specified, `words` wins.

Usage

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

Arguments

`data`	A `data.table` (of new class `wordvec`) loaded by `data_wordvec_load`.
`words`	[Option 1] Word strings (`NULL`; a single word; a vector of words).
`pattern`	[Option 2] Pattern of regular expression (see `str_subset`).
`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

`get_wordvec`

`plot_wordvec`

`plot_wordvec_tSNE`

Examples

```
d = data_wordvec_normalize(demodata)

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

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

## plot word vectors:
get_wordvecs(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_wordvecs(
  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, topn = 10, keep = FALSE, above = NULL, verbose = TRUE)
```

Arguments

`data`	A `data.table` (of new class `wordvec`) loaded by `data_wordvec_load`.

`x`	Can be one of the following:
	• 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`.
`keep`	Keep words specified in `x` in results? Defaults to `FALSE`.
`above`	Defaults to `NULL`. Can be one of the following: • 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.
`verbose`	Print information to the console? Defaults to `TRUE`.

Value

A `data.table` with the most similar words and their cosine similarities. The row number of each word in the raw data is also returned, which may help determine the relative word frequency in some cases.

Two attributes are appended to the returned `data.table` (see examples): `wordvec` and `wordvec.formula`. Users may extract them for further use.

Download

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

See Also

cosine_similarity

pair_similarity

tab_similarity

Examples

```
d = data_wordvec_normalize(demodata)

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)
attr(ms, "dims")
attr(ms, "normalized")
attr(ms, "wordvec.formula")
attr(ms, "wordvec")
# final word vector computed according to the formula
```

`pair_similarity` *Compute cosine similarity/distance for a pair of words.*

Description

Compute cosine similarity/distance for a pair of words.

Usage

```
pair_similarity(data, word1, word2, distance = FALSE)
```

Arguments

data	A data.table (of new class wordvec) loaded by data_wordvec_load.
word1, word2	Word string (a single word).
distance	Compute cosine distance instead? Defaults to FALSE (cosine similarity).

Value

A value of cosine similarity/distance.

Download

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

See Also

cosine_similarity

tab_similarity

most_similar

Examples

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

---

plot_wordvec *Visualize word vectors.*

Description

Visualize word vectors.

Usage

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

Arguments

dt	A data.table returned by get_wordvecs or 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

plot_wordvec_tSNE

Examples

```
d = data_wordvec_normalize(demodata)

dt = get_wordvecs(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_wordvecs(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_wordvecs(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(
  dt,
  dims = 2,
  perplexity,
  theta = 0.5,
  colors = NULL,
  seed = NULL,
  custom.Rtsne = NULL
)
```

Arguments

`dt`	A `data.table` returned by `get_wordvecs` or 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 to obtain 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

Examples

```
d = data_wordvec_normalize(demodata)

dt = get_wordvecs(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)
```

---

tab_similarity *Tabulate data for cosine similarity/distance of all word pairs.*

Description

Tabulate data for cosine similarity/distance of all word pairs.

Usage

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

Arguments

data	A data.table (of new class wordvec) loaded by data_wordvec_load.
words	[Option 1] Word strings (NULL; a single word; a vector of words).
pattern	[Option 2] Pattern of regular expression (see str_subset).
unique	Word pairs: unique pairs (TRUE) or full pairs with duplicates (FALSE; default).
distance	Compute cosine distance instead? Defaults to FALSE (cosine similarity).

Value

A data.table of all words and their unique or full pairs, with 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

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)
```

test_RND *Relative Norm Distance (RND) analysis.*

Description

Tabulate data for the *Relative Norm Distance* (RND; also known as *Relative Euclidean Distance*) analysis. This is an alternative method to Single-Category WEAT.

Usage

```
test_RND(data, T1, A1, A2, use.pattern = FALSE, labels, rev = FALSE)
```

Arguments

data	A data.table (of new class wordvec) loaded by 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").
rev	Reverse the results? Defaults to FALSE, which means that a positive (vs. negative) value of RND indicates the target words are more associated with A2 than A1. If reversed (TRUE), then the interpretation of RND will also be reversed.

Value

A list of objects:

words.valid valid (actually matched) words

data.rnd data.table of (raw and relative) norm distances

code.diff description for the difference between the two attribute concepts

eff.type effect type: RND

eff.sum sum of RND (a single value)

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

test_WEAT

Examples

```
rnd = test_RND(
  demodata,
  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"),
  labels=list(T1="Occupation", A1="Male", A2="Female"))
rnd
```

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

Description

Tabulate data (cosine similarity and standardized effect size) for the *Word Embedding Association Test* (WEAT) and *Single-Category Word Embedding Association Test* (SC-WEAT) analyses.

Usage

```
test_WEAT(data, T1, T2, A1, A2, use.pattern = FALSE, labels)
```

Arguments

`data`	A `data.table` (of new class `wordvec`) loaded by `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")`.

Value

A `list` of objects:

`words.valid` valid (actually matched) words

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

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

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

`code.diff` description for the difference between the two attribute concepts

`eff.type` effect type: WEAT or SC-WEAT

`eff.raw` raw effect for WEAT (a single value) or SC-WEAT (a data table)

`eff.size` standardized effect size for WEAT (a single value) or SC-WEAT (a data table)

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`

`test_RND`

Examples

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

weat = test_WEAT(
  demodata,
  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"),
  labels=list(T1="King", T2="Queen", A1="Male", A2="Female"))
weat

weat = test_WEAT(
  demodata,
```

```
  T1="^[kK]ing$",
  T2="^[qQ]ueen$",
  A1="^male$|^man$|^boy$|^brother$|^he$|^him$|^his$|^son$",
  A2="^female$|^woman$|^girl$|^sister$|^she$|^her$|^hers$|^daughter$",
  use.pattern=TRUE,
  labels=list(T1="King", T2="Queen", A1="Male", A2="Female"))
weat

sc_weat = test_WEAT(
  demodata,
  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"),
  labels=list(T1="Occupation", A1="Male", A2="Female"))
sc_weat
```

`tokenize` *Tokenize raw texts for training word vectors.*

Description

Tokenize raw texts for training word vectors.

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 the text, or a file path on disk containing the 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. 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 word vectors using the Word2Vec, GloVe, or FastText algorithm.*

Description

Train word vectors 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 the text, or a file path on disk containing the 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 down-sampled. 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 `data_wordvec_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 `data.table` (of new class `wordvec`) with two variables:

`word` words (tokens)

`vec` **raw** *or* **normalized** word vectors

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)

as_tibble(dt1)  # just check
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)

as_tibble(dt2)  # just check
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)

as_tibble(dt3)  # just check
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