

Package ‘wordsalad’

October 12, 2022

Title Provide Tools to Extract and Analyze Word Vectors

Version 0.2.0

Description Provides access to various word embedding methods (GloVe, fasttext and word2vec) to extract word vectors using a unified framework to increase reproducibility and correctness.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

Imports tibble, text2vec, word2vec, fastTextR

Suggests testthat

URL <https://github.com/EmilHvitfeldt/wordsalad>

BugReports <https://github.com/EmilHvitfeldt/wordsalad/issues>

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre] (<<https://orcid.org/0000-0002-0679-1945>>)

Maintainer Emil Hvitfeldt <emilhhvitfeldt@gmail.com>

Repository CRAN

Date/Publication 2020-09-23 08:10:03 UTC

R topics documented:

fairy_tales	2
fasttext	2
glove	4
word2vec	5
Index	8

fairy_tales

The text of H.C. Andersen's fairy tales in English

Description

A dataset containing 5 of H.C. andersens fairy tales translated to English. The UTF-8 plain text was sourced from <http://www.andersenstories.com/>.

Usage

```
fairy_tales
```

Format

A character vector with 5 elements.

Details

This is not representative of the size needed to generate good word vectors. It is just used for examples.

fasttext

Extract word vectors from fasttext word embedding

Description

The calculations are done with the fastTextR package.

Usage

```
fasttext(  
  text,  
  tokenizer = text2vec::space_tokenizer,  
  dim = 10L,  
  type = c("skip-gram", "cbow"),  
  window = 5L,  
  loss = "hs",  
  negative = 5L,  
  n_iter = 5L,  
  min_count = 5L,  
  threads = 1L,  
  composition = c("tibble", "data.frame", "matrix"),  
  verbose = FALSE  
)
```

Arguments

text	Character string.
tokenizer	Function, function to perform tokenization. Defaults to text2vec::space_tokenizer .
dim	Integer, number of dimension of the resulting word vectors.
type	Character, the type of algorithm to use, either 'cbow' or 'skip-gram'. Defaults to 'skip-gram'.
window	Integer, skip length between words. Defaults to 5.
loss	Character, choice of loss function must be one of "ns", "hs", or "softmax". See details for more Defaults to "hs".
negative	integer with the number of negative samples. Only used when loss = "ns".
n_iter	Integer, number of training iterations. Defaults to 5. <code>numeric = -1</code> defines early stopping strategy. Stop fitting when one of two following conditions will be satisfied: (a) passed all iterations (b) $\text{cost_previous_iter} / \text{cost_current_iter} - 1 < \text{convergence_tol}$. Defaults to -1.
min_count	Integer, number of times a token should appear to be considered in the model. Defaults to 5.
threads	number of CPU threads to use. Defaults to 1.
composition	Character, Either "tibble", "matrix", or "data.frame" for the format out the resulting word vectors.
verbose	Logical, controls whether progress is reported as operations are executed.

Details

The choice of loss functions are one of:

- "ns" negative sampling
- "hs" hierarchical softmax
- "softmax" full softmax

Value

A [tibble](#), `data.frame` or matrix containing the token in the first column and word vectors in the remaining columns.

Source

<https://fasttext.cc/>

References

Enriching Word Vectors with Subword Information, 2016, P. Bojanowski, E. Grave, A. Joulin, T. Mikolov.

Examples

```
fasttext(fairy_tales, n_iter = 2)

# Custom tokenizer that splits on non-alphanumeric characters
fasttext(fairy_tales,
         n_iter = 2,
         tokenizer = function(x) strsplit(x, "[^[:alnum:]]+"))
```

glove

*Extract word vectors from GloVe word embedding***Description**

The calculations are done with the `text2vec` package.

Usage

```
glove(
  text,
  tokenizer = text2vec::space_tokenizer,
  dim = 10L,
  window = 5L,
  min_count = 5L,
  n_iter = 10L,
  x_max = 10L,
  stopwords = character(),
  convergence_tol = -1,
  threads = 1,
  composition = c("tibble", "data.frame", "matrix"),
  verbose = FALSE
)
```

Arguments

<code>text</code>	Character string.
<code>tokenizer</code>	Function, function to perform tokenization. Defaults to <code>text2vec::space_tokenizer</code> .
<code>dim</code>	Integer, number of dimension of the resulting word vectors.
<code>window</code>	Integer, skip length between words. Defaults to 5.
<code>min_count</code>	Integer, number of times a token should appear to be considered in the model. Defaults to 5.
<code>n_iter</code>	Integer, number of training iterations. Defaults to 10.
<code>x_max</code>	Integer, maximum number of co-occurrences to use in the weighting function. Defaults to 10.
<code>stopwords</code>	Character, a vector of stop words to exclude from training.

convergence_tol	Numeric, value determining the convergence criteria. <code>numeric = -1</code> defines early stopping strategy. Stop fitting when one of two following conditions will be satisfied: (a) passed all iterations (b) $\text{cost_previous_iter} / \text{cost_current_iter} - 1 < \text{convergence_tol}$. Defaults to -1.
threads	number of CPU threads to use. Defaults to 1.
composition	Character, Either "tibble", "matrix", or "data.frame" for the format out the resulting word vectors.
verbose	Logical, controls whether progress is reported as operations are executed.

Value

A [tibble](#), `data.frame` or matrix containing the token in the first column and word vectors in the remaining columns.

Source

<https://nlp.stanford.edu/projects/glove/>

References

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Examples

```
glove(fairy_tales, x_max = 5)
```

word2vec

Extract word vectors from word2vec word embedding

Description

The calculations are done with the `word2vec` package.

Usage

```
word2vec(  
  text,  
  tokenizer = text2vec::space_tokenizer,  
  dim = 50,  
  type = c("cbow", "skip-gram"),  
  window = 5L,  
  min_count = 5L,  
  loss = c("ns", "hs"),  
  negative = 5L,  
  n_iter = 5L,  
)
```

```

  lr = 0.05,
  sample = 0.001,
  stopwords = character(),
  threads = 1L,
  collapse_character = "\t",
  composition = c("tibble", "data.frame", "matrix")
)

```

Arguments

<code>text</code>	Character string.
<code>tokenizer</code>	Function, function to perform tokenization. Defaults to <code>text2vec::space_tokenizer</code> .
<code>dim</code>	dimension of the word vectors. Defaults to 50.
<code>type</code>	the type of algorithm to use, either 'cbow' or 'skip-gram'. Defaults to 'cbow'
<code>window</code>	skip length between words. Defaults to 5.
<code>min_count</code>	integer indicating the number of time a word should occur to be considered as part of the training vocabulary. Defaults to 5.
<code>loss</code>	Character, choice of loss function must be one of "ns" or "hs". See details for more Defaults to "ns".
<code>negative</code>	integer with the number of negative samples. Only used in case hs is set to FALSE
<code>n_iter</code>	Integer, number of training iterations. Defaults to 5.
<code>lr</code>	initial learning rate also known as alpha. Defaults to 0.05
<code>sample</code>	threshold for occurrence of words. Defaults to 0.001
<code>stopwords</code>	a character vector of stopwords to exclude from training
<code>threads</code>	number of CPU threads to use. Defaults to 1.
<code>collapse_character</code>	Character vector with length 1. Character used to glue together tokens after tokenizing. See details for more information. Defaults to "\t".
<code>composition</code>	Character, Either "tibble", "matrix", or "data.frame" for the format out the resulting word vectors.

Details

A trade-off have been made to allow for an arbitrary tokenizing function. The text is first passed through the tokenizer. Then it is being collapsed back together into strings using `collapse_character` as the separator. You need to pick `collapse_character` to be a character that will not appear in any of the tokens after tokenizing is done. The default value is a "tab" character. If you pick a character that is present in the tokens then those words will be split.

The choice of loss functions are one of:

- "ns" negative sampling
- "hs" hierarchical softmax

Value

A [tibble](#), data.frame or matrix containing the token in the first column and word vectors in the remaining columns.

Source

<https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>

References

Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff. 2013. Distributed Representations of Words and Phrases and their Compositionality

Examples

```
word2vec(fairy_tales)

# Custom tokenizer that splits on non-alphanumeric characters
word2vec(fairy_tales, tokenizer = function(x) strsplit(x, "[^[:alnum:]]+"))
```

Index

* datasets

fairy_tales, 2

fairy_tales, 2

fasttext, 2

glove, 4

text2vec::space_tokenizer, 3, 4, 6

tibble, 3, 5, 7

word2vec, 5