Package ‘conText’

October 25, 2021

Version 1.0.0

Title 'a la Carte' on Text (ConText) Embedding Regression


License GPL-3

Depends R (>= 3.6.0)

Imports dplyr, Matrix (>= 1.3-2), quanteda (>= 3.0.0), text2vec (>= 0.6), reshape2 (>= 1.4.4), fastDummies (>= 1.6.3), stringr (>= 1.4.0), ggplot2, methods

URL https://github.com/prodriguezsosa/EmbeddingRegression

BugReports https://github.com/prodriguezsosa/ConText/issues

Maintainer Pedro L. Rodriguez <pedro.rodriguez@Vanderbilt.Edu>

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Language en-US

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

NeedsCompilation no

Author Pedro L. Rodriguez [aut, cre, cph]

Arthur Spirling [aut] (<https://orcid.org/0000-0001-9959-1805>), Brandon Stewart [aut] (<https://orcid.org/0000-0002-7657-3089>), Christopher Barrie [ctb] (<https://orcid.org/0000-0002-9156-990X>)

Repository CRAN

Date/Publication 2021-10-25 07:00:16 UTC
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>bootstrap_contrast</td>
<td>3</td>
</tr>
<tr>
<td>bootstrap_nns</td>
<td>3</td>
</tr>
<tr>
<td>bootstrap_ols</td>
<td>5</td>
</tr>
<tr>
<td>bootstrap_similarity</td>
<td>6</td>
</tr>
<tr>
<td>build_conText</td>
<td>6</td>
</tr>
<tr>
<td>build_dem</td>
<td>7</td>
</tr>
<tr>
<td>build_fem</td>
<td>8</td>
</tr>
<tr>
<td>compute_contrast</td>
<td>8</td>
</tr>
<tr>
<td>compute_similarity</td>
<td>9</td>
</tr>
<tr>
<td>compute_transform</td>
<td>10</td>
</tr>
<tr>
<td>conText</td>
<td>11</td>
</tr>
<tr>
<td>contrast_nns</td>
<td>13</td>
</tr>
<tr>
<td>cos_sim</td>
<td>15</td>
</tr>
<tr>
<td>cr_glove_subset</td>
<td>16</td>
</tr>
<tr>
<td>cr_sample_corpus</td>
<td>17</td>
</tr>
<tr>
<td>cr_transform</td>
<td>17</td>
</tr>
<tr>
<td>dem</td>
<td>18</td>
</tr>
<tr>
<td>dem_group</td>
<td>19</td>
</tr>
<tr>
<td>embed_target</td>
<td>20</td>
</tr>
<tr>
<td>feature_sim</td>
<td>21</td>
</tr>
<tr>
<td>fem</td>
<td>22</td>
</tr>
<tr>
<td>find_cos_sim</td>
<td>23</td>
</tr>
<tr>
<td>find_nns</td>
<td>24</td>
</tr>
<tr>
<td>get_context</td>
<td>24</td>
</tr>
<tr>
<td>get_cos_sim</td>
<td>26</td>
</tr>
<tr>
<td>get_local_vocab</td>
<td>27</td>
</tr>
<tr>
<td>get_nns</td>
<td>28</td>
</tr>
<tr>
<td>get_nns_ratio</td>
<td>30</td>
</tr>
<tr>
<td>get_seq_cos_sim</td>
<td>32</td>
</tr>
<tr>
<td>nns</td>
<td>34</td>
</tr>
<tr>
<td>nns_ratio</td>
<td>35</td>
</tr>
<tr>
<td>permute_contrast</td>
<td>37</td>
</tr>
<tr>
<td>permute_ols</td>
<td>37</td>
</tr>
<tr>
<td>plot_nns_ratio</td>
<td>38</td>
</tr>
<tr>
<td>prototypical_context</td>
<td>39</td>
</tr>
<tr>
<td>run_ols</td>
<td>40</td>
</tr>
<tr>
<td>tokens_context</td>
<td>41</td>
</tr>
</tbody>
</table>
bootstrap_contrast

---

**Description**

Bootstrap similarity and ratio computations

**Usage**

```r
bootstrap_contrast(
  target_embeddings1 = NULL,
  target_embeddings2 = NULL,
  pre_trained = NULL,
  candidates = NULL,
  norm = NULL
)
```

**Arguments**

- `target_embeddings1`: ALC embeddings for group 1
- `target_embeddings2`: ALC embeddings for group 2
- `pre_trained`: a V x D matrix of numeric values - pretrained embeddings with V = size of vocabulary and D = embedding dimensions
- `candidates`: character vector defining the candidates for nearest neighbors - e.g. output from `get_local_vocab`
- `norm`: character = c("l2", "none") - set to 'l2' for cosine similarity and to 'none' for inner product (see ?sim2 in text2vec)

**Value**

a list with three elements, nns for group 1, nns for group 2 and nns_ratio comparing with ratios of similarities between the two groups

---

bootstrap_nns

---

**Description**

Uses bootstrapping–sampling of texts with replacement– to identify the top N nearest neighbors based on cosine or inner product similarity.
Usage

```r
bootstrap_nns(
  context = NULL,
  pre_trained = NULL,
  transform = TRUE,
  transform_matrix = NULL,
  candidates = NULL,
  bootstrap = TRUE,
  num_bootstraps = 20,
  N = 50,
  norm = "l2"
)
```

Arguments

- **context**: (character) vector of texts - context variable in `get_context` output
- **pre_trained**: (numeric) a F x D matrix corresponding to pre-trained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
- **transform**: (logical) - if TRUE (default) apply the a la carte transformation, if FALSE output untransformed averaged embedding.
- **transform_matrix**: (numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pre-trained embeddings.
- **candidates**: (character) vector defining the candidates for nearest neighbors - e.g. output from `get_local_vocab`.
- **bootstrap**: (logical) if TRUE, bootstrap similarity values - sample from texts with replacement. Required to get std. errors.
- **num_bootstraps**: (numeric) - number of bootstraps to use.
- **N**: (numeric) number of nearest neighbors to return.
- **norm**: (character) - how to compute the similarity (see `?text2vec::sim2`):
  - "l2" cosine similarity
  - "none" inner product

Value

A `data.frame` with the following columns:

- **feature**: (character) vector of feature terms corresponding to the nearest neighbors.
- **value**: (numeric) cosine/inner product similarity between texts and feature. Average over bootstrapped samples if bootstrap = TRUE.

std.error (numeric) std. error of the similarity value. Column is dropped if bootstrap = FALSE.
Examples

# find contexts of immigration
collection_immigration <- get_context(x = cr_sample_corpus, 
  target = 'immigration',
  window = 6,
  valuetype = "fixed",
  case_insensitive = TRUE,
  hard_cut = FALSE, verbose = FALSE)

# find local vocab (use it to define the candidate of nearest neighbors)
local_vocab <- get_local_vocab(collection_immigration$context, pre_trained = cr_glove_subset)

set.seed(42L)
nns_immigration <- bootstrap_nns(collection = collection_immigration$context, 
  pre_trained = cr_glove_subset,
  transform_matrix = cr_transform, 
  transform = TRUE, 
  candidates = local_vocab, 
  bootstrap = TRUE, 
  num_bootstraps = 20, N = 50, 
  norm = "l2")

---

**bootstrap_ols**  
**Bootstrap OLS**

Description

Bootstrap model coefficients and standard errors

Usage

`bootstrap_ols(Y = NULL, X = NULL, stratify = NULL)`

Arguments

- **Y**  
  vector of regression model’s dependent variable (embedded context)
- **X**  
  data.frame of model independent variables (covariates)
- **stratify**  
  covariates to stratify when bootstrapping

Value

list with two elements, betas = list of beta_coefficients (D dimensional vectors); normed_betas = tibble with the norm of the non-intercept coefficients
bootstrap_similarity  

*Bootstrap similarity vector*

**Description**

Bootstrap similarity vector

**Usage**

```r
bootstrap_similarity(
  target_embeddings = NULL,
  pre_trained = NULL,
  candidates = NULL,
  norm = NULL
)
```

**Arguments**

- `target_embeddings`  
  the target embeddings (embeddings of context)
- `pre_trained`  
  a V x D matrix of numeric values - pretrained embeddings with V = size of vocabulary and D = embedding dimensions
- `candidates`  
  character vector defining the candidates for nearest neighbors - e.g. output from `get_local_vocab`
- `norm`  
  character = c("l2", "none") - set to 'l2' for cosine similarity and to 'none' for inner product (see ?sim2 in text2vec)

**Value**

vector(s) of cosine similarities between alc embedding and nearest neighbor candidates

---

**build_conText**  

*build a conText-class object*

**Description**

build a conText-class object

**Usage**

```r
build_conText(
  Class = "conText",
  x_conText,
  normed_coefficients = data.frame(),
  features = character(),
  Dimnames = list()
)
```
build_dem

Arguments

- **Class**: defines the class of this object (fixed)
- **x_conText**: a dgCMatrix class matrix
- **normed_coefficients**: a data.frame with the normed coefficients and other statistics
- **features**: features used in computing the embeddings
- **Dimnames**: row (features) and columns (NULL) names

Description

build a dem-class object

Usage

```r
build_dem(
  Class = "em",
  x_dem,
  docvars = data.frame(),
  features = character(),
  Dimnames = list()
)
```

Arguments

- **Class**: defines the class of this object (fixed)
- **x_dem**: a dgCMatrix class matrix
- **docvars**: document covariates, inherited from dfm and corpus, subset to embeddable documents
- **features**: features used in computing the embeddings
- **Dimnames**: row (documents) and columns (NULL) names
build_fem  
build a fem-class object

Description

build a fem-class object

Usage

build_fem(
  Class = "fem",
  x_fem,
  features = character(),
  counts = numeric(),
  Dimnames = list()
)

Arguments

Class  defines the class of this object (fixed)

x_fem  a dgCMatrix class matrix

features  features used in computing the embeddings

counts  counts of features used in computing embeddings

Dimnames  row (features) and columns (NULL) names

compute_contrast  
Compute similarity and similarity ratios

Description

Compute similarity and similarity ratios

Usage

compute_contrast(
  target_embeddings1 = NULL,
  target_embeddings2 = NULL,
  pre_trained = NULL,
  candidates = NULL,
  norm = NULL
)
**compute_similarity**

*Compute similarity vector (sub-function of bootstrap_similarity)*

**Description**

Compute similarity vector (sub-function of bootstrap_similarity)

**Usage**

```
compute_similarity(
  target_embeddings = NULL,
  pre_trained = NULL,
  candidates = NULL,
  norm = NULL
)
```

**Arguments**

- **target_embeddings**: the target embeddings (embeddings of context)
- **pre_trained**: a V x D matrix of numeric values - pretrained embeddings with V = size of vocabulary and D = embedding dimensions
- **candidates**: character vector defining the candidates for nearest neighbors - e.g. output from `get_local_vocab`
- **norm**: character = c("l2", "none") - set to 'l2' for cosine similarity and to 'none' for inner product (see ?sim2 in text2vec)

**Value**

vector of cosine similarities between alc embedding and nearest neighbor candidates
compute_transform  Compute transformation matrix A

Description

Computes a transformation matrix, given a feature-co-occurrence matrix and corresponding pre-trained embeddings.

Usage

compute_transform(x, pre_trained, weighting = 500)

Arguments

x  a (quanteda) fcm-class object.
pre_trained  (numeric) a F x D matrix corresponding to pretrained embeddings, usually trained on the same corpus as that used for x. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding
weighting  (character or numeric) weighting options:
1 no weighting.
"log" weight by the log of the frequency count.
numeric threshold based weighting (= 1 if token count meets threshold, 0 ow).
Recommended: use log for small corpora, a numeric threshold for larger corpora.

Value

a dgTMatrix-class D x D non-symmetrical matrix (D = dimensions of pre-trained embedding space) corresponding to an 'a la carte' transformation matrix. This matrix is optimized for the corpus and pre-trained embeddings employed.

Examples

library(quanteda)

# note, cr_sample_corpus is too small to produce sensical word vectors

# tokenize
toks <- tokens(cr_sample_corpus)

# construct feature-co-occurrence matrix
toks_fcm <- fcm(toks, context = "window", window = 6,
count = "weighted", weights = 1 / (1:6), tri = FALSE)

# you will generally want to estimate a new (corpus-specific)
# GloVe model, we will use cr_glove_subset instead
# see the Quick Start Guide to see a full example.

# estimate transform
local_transform <- compute_transform(x = toks_fcm,
    pre_trained = cr_glove_subset, weighting = 'log')

---

**conText**

**Embedding regression**

**Description**

Estimates an embedding regression model with options to use bootstrapping to estimate confidence
intervals and a permutation test for inference (see https://github.com/prodriguezsosa/conText for
details.)

**Usage**

```r
conText(
    formula, data,
    pre_trained,
    transform = TRUE,
    transform_matrix,
    bootstrap = TRUE,
    num_bootstraps = 20,
    stratify = TRUE,
    permute = TRUE,
    num_permutations = 100,
    window = 6L,
    valuetype = c("glob", "regex", "fixed"),
    case_insensitive = TRUE,
    hard_cut = FALSE,
    verbose = TRUE
)
```

**Arguments**

- **formula**
  - a symbolic description of the model to be fitted with a target word as a DV
e.g. immigrant ~ party + gender. To use a phrase as a DV, place it quotations
e.g. "immigrant refugees" ~ party + gender. To use all covariates included
  in the data, you can use . on RHS, e.g.immigrant ~ . If you wish to treat the
  full document as you DV, rather than a single target word, use . on the LHS e.g.
  . ~ party + gender.

- **data**
  - a quanteda tokens-class object with the necessary document variables. Co-
    variates must be either binary indicator variables or "transformable" into binary
indicator variables. conText will automatically transform any non-indicator variables into binary indicator variables (multiple if more than 2 classes), leaving out a "base" category.

**pre_trained**
(numeric) a \( F \times D \) matrix corresponding to pretrained embeddings. \( F = \) number of features and \( D = \) embedding dimensions. \( \text{rownames(pre_trained)} = \) set of features for which there is a pre-trained embedding.

**transform**
(logical) if TRUE (default) apply the 'a la carte' transformation, if FALSE output untransformed averaged embeddings.

**transform_matrix**
(numeric) a \( D \times D \) 'a la carte' transformation matrix. \( D = \) dimensions of pretrained embeddings.

**bootstrap**
(logical) if TRUE, use bootstrapping – sample from texts with replacement and re-run regression on each sample. Required to get std. errors.

**num_bootstraps**
(numeric) number of bootstraps to use

**stratify**
(logical) if TRUE, stratify by covariates when bootstrapping

**permute**
(logical) if TRUE, compute empirical p-values using permutation test

**num_permutations**
(numeric) number of permutations to use

**window**
the number of context words to be displayed around the keyword

**valuetype**
the type of pattern matching: "glob" for "glob"-style wildcard expressions; "regex" for regular expressions; or "fixed" for exact matching. See valuetype for details.

**case_insensitive**
logical; if TRUE, ignore case when matching a pattern or dictionary values

**hard_cut**
(logical) - if TRUE then a context must have \( \text{window} \times 2 \) tokens, if FALSE it can have \( \text{window} \times 2 \) or fewer (e.g. if a doc begins with a target word, then context will have \( \text{window} \) tokens rather than \( \text{window} \times 2 \))

**verbose**
(logical) - if TRUE, report the documents that had no overlapping features with the pretrained embeddings provided.

**Value**

a conText-class object - a \( D \times M \) matrix with \( D = \) dimensions of the pre-trained feature embeddings provided and \( M = \) number of covariates including the intercept. These represent the estimated regression coefficients. These can be combined to compute ALC embeddings for different combinations of covariates. The object also includes various informative attributes, importantly a data.frame with the following columns:

**coefficient**
(character) name of (covariate) coefficient.

**value**
(numeric) norm of the corresponding beta coefficient.

**std.error**
(numeric) (if bootstrap = TRUE) std. error of the norm of the beta coefficient.

**p.value**
(numeric) (if permute = TRUE) empirical p.value of the norm of the coefficient.
Examples

library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr\*", window = 6L)

## given the target word "immigration"
set.seed(2021L)
model1 <- conText(formula = immigration ~ party + gender,
data = toks,
pre_trained = cr_glove_subset,
transform = TRUE, transform_matrix = cr_transform,
bootstrap = TRUE, num_bootstraps = 10,
stratify = TRUE,
permute = TRUE, num_permutations = 100,
window = 6, case_insensitive = TRUE,
verbose = FALSE)

# notice, non-binary covariates are automatically "dummified"
rownames(model1)

# the beta coefficient 'partyR' in this case corresponds to the alc embedding
# of "immigration" for Republican party speeches

# (normed) coefficient table
model1@normed_coefficients

---

contrast_nns

**Contrast nearest neighbors**

**Description**

Computes the ratio of cosine similarities between group embeddings and features – that is, for any given feature it first computes the similarity between that feature and each group embedding, and then takes the ratio of these two similarities. This ratio captures how "discriminant" a feature is of a given group.

**Usage**

contrast_nns(
  x,
  groups = NULL,
  pre_trained = NULL,
  transform = TRUE,
transform_matrix = NULL,
bootstrap = TRUE,
um_bootstraps = 20,
permute = TRUE,
um_permutations = 100,
candidates = NULL,
N = 20,
verbose = TRUE )

Arguments

x (quanteda) tokens-class object
groups (numeric, factor, character) a binary variable of the same length as x
pre_trained (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
transform (logical) if TRUE (default) apply the 'a la carte' transformation, if FALSE ouput untransformed averaged embeddings.
transform_matrix (numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pre-trained embeddings.
bootstrap (logical) if TRUE, use bootstrapping – sample from texts with replacement and re-estimate cosine ratios for each sample. Required to get std. errors.
um_bootstraps (numeric) - number of bootstraps to use
permute (logical) - if TRUE, compute empirical p-values using a permutation test
num_permutations (numeric) - number of permutations to use
candidates (character) vector of candidate features for nearest neighbors
N (numeric) - nearest neighbors are subset to the union of the N neighbors of each group (if NULL, ratio is computed for all features)
verbose (logical) - if TRUE, report the documents that had no overlapping features with the pretrained embeddings provided.

Value

da data.frame with following columns:

feature (character) vector of feature terms corresponding to the nearest neighbors.
value (numeric) ratio of cosine similarities. Average over bootstrapped samples if bootstrap = TRUE.
std.error (numeric) std. error of the ratio of cosine similarities. Column is dropped if bootstrap = FALSE.
p.value (numeric) empirical p-value. Column is dropped if permute = FALSE.
Examples

```r
library(quanteda)

cr_toks <- tokens(cr_sample_corpus)

immig_toks <- tokens_context(x = cr_toks,
  pattern = "immigration", window = 6L, hard_cut = FALSE, verbose = TRUE)

set.seed(42L)
party_nns <- contrast_nns(x = immig_toks,
  groups = docvars(immig_toks, 'party'),
  pre_trained = cr_glove_subset,
  transform = TRUE, transform_matrix = cr_transform,
  bootstrap = TRUE, num_bootstraps = 10,
  permute = TRUE, num_permutations = 100,
  candidates = NULL, N = 20,
  verbose = FALSE)
```

---

**cos_sim**

Compute the cosine similarity between one or more ALC embeddings and a set of features.

### Description

Compute the cosine similarity between one or more ALC embeddings and a set of features.

### Usage

```r
cos_sim(x, pre_trained, features = NULL, as_list = TRUE)
```

### Arguments

- `x` a (quanteda) dem-class or fem-class object.
- `pre_trained` (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
- `features` (character) features of interest.
- `as_list` (logical) if FALSE all results are combined into a single data.frame If TRUE, a list of data.frames is returned with one data.frame per feature.

### Value

A data.frame or list of data.frames (one for each target) with the following columns:

- `target` (character) rownames of x, the labels of the ALC embeddings.
- `feature` (character) feature terms defined in the features argument.
- `value` (numeric) cosine similarity between x and feature.
Examples

```r
library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)

# build document-feature matrix
immig_dfm <- dfm(immig_toks)

# construct document-embedding-matrix
immig_dem <- dem(immig_dfm, pre_trained = cr_glove_subset,
transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)

# to get group-specific embeddings, average within party
immig_wv_party <- dem_group(immig_dem, groups = immig_dem@docvars$party)

# compute the cosine similarity between each party’s embedding and a specific set of features
cos_sim(immig_wv_party, pre_trained = cr_glove_subset,
features = c("reform", "enforcement"), as_list = FALSE)
```

---

**cr_glove_subset**  
**GloVe subset**

**Description**

A subset of a GloVe embeddings model trained on the top 5000 features in the Congressional Record corpus covering the 111th - 114th Congresses, and limited to speeches by Democrat and Republican representatives.

**Usage**

`cr_glove_subset`

**Format**

A matrix with 500 rows and 300 columns:

- **row** each row corresponds to a word
- **column** each column corresponds to a dimension in the embedding space

**Source**

**cr_sample_corpus**

**Congressional Record sample corpus**

**Description**

A (quanteda) corpus containing a sample of the United States Congressional Record (daily transcripts) covering the 111th to 114th Congresses. The raw corpus is first subset to speeches containing the regular expression "immig*". Then 100 docs from each party-gender pair is randomly sampled. For full data and pre-processing file, see: https://www.dropbox.com/sh/jsyrag7opfo7l7i/AAB1z7tumLuKihGu2-FDmhmKa?dl=0

**Usage**

`cr_sample_corpus`

**Format**

A quanteda corpus with 200 documents and 3 docvars:

- **party** party of speaker, (D)emocrat or (R)epublican
- **gender** gender of speaker, (F)emale or (M)ale
- **session_id** id of Congress session in which speech was given ...

**Source**

[https://data.stanford.edu/congress_text](https://data.stanford.edu/congress_text)

---

**cr_transform**

**Transformation matrix**

**Description**

A square matrix corresponding to the transformation matrix computed using the cr_glove_subset embeddings and corresponding corpus.

**Usage**

`cr_transform`

**Format**

A 300 by 300 matrix.

**Source**

**Build a document-embedding matrix**

**Description**

Given a document-feature-matrix, for each document, multiply its feature counts (columns) with their corresponding pretrained word embeddings and average (usually referred to as averaged or additive document embeddings). If specified and a transformation matrix is provided, multiply the document embeddings by the transformation matrix to obtain the corresponding a la carte document embeddings. (see eq 2: https://arxiv.org/pdf/1805.05388.pdf)

**Usage**

dem(x, pre_trained, transform = TRUE, transform_matrix, verbose = TRUE)

**Arguments**

- **x**: a quanteda (dfm-class) document-feature-matrix
- **pre_trained**: (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
- **transform**: (logical) if TRUE (default) apply the 'a la carte' transformation, if FALSE output untransformed averaged embeddings.
- **transform_matrix**: (numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pretrained embeddings.
- **verbose**: (logical) - if TRUE, report the documents that had no overlapping features with the pretrained embeddings provided.

**Value**

A N x D (dem-class) document-embedding-matrix corresponding to the ALC embeddings for each document. N = number of documents (that could be embedded), D = dimensions of pretrained embeddings. This object inherits the document variables in x, the dfm used. These can be accessed calling the attribute: @docvars. Note, documents with no overlapping features with the pretrained embeddings provided are automatically dropped. For a list of the documents that were embedded call the attribute: @Dimnames$docs.

**Examples**

```r
library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
```
```
dem_group

immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)

# construct document-feature-matrix
immig_dfm <- dfm(immig_toks)

# construct document-embedding-matrix
immig_dem <- dem(immig_dfm, pre_trained = cr_glove_subset,
                  transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)
```

---

### dem_group

**Average document-embeddings in a dem by a grouping variable**

**Description**

Average embeddings in a dem by a grouping variable, by averaging over columns within groups and creating new "documents" with the group labels. Similar in essence to `dfm_group`.

**Usage**

```
dem_group(x, groups = NULL)
```

**Arguments**

- `x` a `dem-class` document-embedding-matrix
- `groups` a character or factor variable equal in length to the number of documents

**Value**

A `G x D` (`dem-class`) document-embedding-matrix corresponding to the ALC embeddings for each group. `G` = number of unique groups defined in the `groups` variable, `D` = dimensions of pretrained embeddings.

**Examples**

```
library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)

# build document-feature matrix
immig_dfm <- dfm(immig_toks)

# construct document-embedding-matrix
immig_dem <- dem(immig_dfm, pre_trained = cr_glove_subset,
                 transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)
```
# to get group-specific embeddings, average within party
immig_wv_party <- dem_group(immig_dem,
groups = immig_dem@docvars$party)

embed_target

Embed target using either: (a) a la carte OR (b) simple (untransformed) averaging of context embeddings

Description

For a vector of contexts (generally the context variable in get_context output), return the transformed (or untransformed) additive embeddings, aggregated or by instance, along with the local vocabulary. Keep track of which contexts were embedded and which were excluded.

Usage

embed_target(
  context,            # (character) vector of texts - context variable in get_context output
  pre_trained,       # (numeric) a F x D matrix corresponding to pretrained embeddings. F = number
                      # of features and D = embedding dimensions. rownames(pre_trained) = set of
                      # features for which there is a pre-trained embedding.
  transform = TRUE,  # (logical) if TRUE (default) apply the ’a la carte’ transformation, if FALSE output
                      # untransformed averaged embeddings.
  transform_matrix,  # (numeric) a D x D ’a la carte’ transformation matrix. D = dimensions of pre-
                      # trained embeddings.
  aggregate = TRUE,  # (logical) - if TRUE (default) output will return one embedding (i.e. averaged
                      # over all instances of target) if FALSE output will return one embedding per
                      # instance
  verbose = TRUE     # (logical) - report the observations that had no overlap the provided pre-trained
                      # embeddings
)

Arguments

context (character) vector of texts - context variable in get_context output
pre_trained (numeric) a F x D matrix corresponding to pretrained embeddings. F = number
of features and D = embedding dimensions. rownames(pre_trained) = set of
features for which there is a pre-trained embedding.
transform (logical) if TRUE (default) apply the ’a la carte’ transformation, if FALSE output
untransformed averaged embeddings.
transform_matrix (numeric) a D x D ’a la carte’ transformation matrix. D = dimensions of pre-
trained embeddings.
aggregate (logical) - if TRUE (default) output will return one embedding (i.e. averaged
over all instances of target) if FALSE output will return one embedding per
instance
verbose (logical) - report the observations that had no overlap the provided pre-trained
embeddings

Details

required packages: quanteda
Value

- list with three elements:
  - target_embedding: the target embedding(s). Values and dimensions will vary with the above settings.
  - local_vocab: (character) vocabulary that appears in the set of contexts provided.
  - obs_included: (integer) rows of the context vector that were included in the computation. A row (context) is excluded when none of the words in the context are present in the pre-trained embeddings provided.

Examples

```r
# find contexts for term immigration
cr_glove_subset <- get_context(x = cr_sample_corpus, target = 'immigration',
    window = 6, valuetype = "fixed", case_insensitive = TRUE,
    hard_cut = FALSE, verbose = FALSE)

cr_glove_subset <- embed_target(context = cr_glove_subset$context,
    pre_trained = cr_glove_subset,
    transform = TRUE, transform_matrix = cr_transform,
    aggregate = FALSE, verbose = FALSE)
```

Description

Efficient way of comparing two corpora along many features simultaneously.

Usage

```r
feature_sim(x, y, features = character(0))
```

Arguments

- `x`: a (fem-class) feature embedding matrix.
- `y`: a (fem-class) feature embedding matrix.
- `features`: (character) vector of features for which to compute similarity scores. If not defined then all overlapping features will be used.

Value

a data.frame with following columns:

- `feature`: (character) overlapping features
- `value`: (numeric) cosine similarity between overlapping features.
Examples

```r
library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# create feature co-occurrence matrix for each party (set tri = FALSE to work with fem)
fcm_D <- fcm(toks[docvars(toks, 'party') == "D"],
context = "window", window = 6, count = "frequency", tri = FALSE)
fcm_R <- fcm(toks[docvars(toks, 'party') == "R"],
context = "window", window = 6, count = "frequency", tri = FALSE)

# compute feature-embedding matrix
fem_D <- fem(fcm_D, pre_trained = cr_glove_subset,
transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)
fem_R <- fem(fcm_R, pre_trained = cr_glove_subset,
transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)

# compare "horizontal" cosine similarity
feat_comp <- feature_sim(x = fem_R, y = fem_D)
```

---

**fem**  
Create an feature-embedding matrix

**Description**

Given a feature co-occurrence matrix for each feature, multiply its feature counts (columns) with their corresponding pre-trained embeddings and average (usually referred to as averaged or additive embeddings). If specified and a transformation matrix is provided, multiply the feature embeddings by the transformation matrix to obtain the corresponding a la carte embeddings. (see eq 2: https://arxiv.org/pdf/1805.05388.pdf)

**Usage**

```r
fem(x, pre_trained, transform = TRUE, transform_matrix, verbose = TRUE)
```

**Arguments**

- **x**  
a quanteda (fcm-class) feature-co-occurrence-matrix
- **pre_trained**  
(numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
- **transform**  
(logical) if TRUE (default) apply the 'a la carte' transformation, if FALSE output untransformed averaged embeddings.
- **transform_matrix**  
(numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pre-trained embeddings.
find_cos_sim

verbose (logical) - if TRUE, report the features that had no overlapping (co-occurring) features with the pretrained embeddings provided.

Value

a fem-class object

Examples

library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# create feature co-occurrence matrix (set tri = FALSE to work with fem)
toks_fcm <- fcm(toks, context = "window", window = 6, count = "frequency", tri = FALSE)

# compute feature-embedding matrix
toks_fem <- fem(toks_fcm, pre_trained = cr_glove_subset, transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)

find_cos_sim  Find cosine similarities between target and candidate words

Description

Find cosine similarities between target and candidate words

Usage

find_cos_sim(target_embedding, pre_trained, candidates, norm = "l2")

Arguments

target_embedding matrix of numeric values
pre_trained matrix of numeric values - pretrained embeddings
candidates character vector defining vocabulary to subset comparison to
norm character = c("l2", "none") - how to scale input matrices. If they are already scaled - use "none" (see ?sim)

Value

a vector of cosine similarities of length candidates
find_nns

Return nearest neighbors based on cosine similarity

Description

Return nearest neighbors based on cosine similarity

Usage

find_nns(target_embedding, pre_trained, N = 5, candidates = NULL, norm = "l2")

Arguments

target_embedding
  (numeric) 1 x D matrix. D = dimensions of pretrained embeddings.

pre_trained
  (numeric) a F x D matrix corresponding to pretrained embeddings. F = number
  of features and D = embedding dimensions. rownames(pre_trained) = set of
  features for which there is a pre-trained embedding.

N
  (numeric) number of nearest neighbors to return.

candidates
  (character) vector of candidate features for nearest neighbors

norm
  (character) - how to compute similarity (see ?text2vec::sim2):
  "l2" cosine similarity
  "none" inner product

Value

(character) vector of nearest neighbors to target

Examples

find_nns(target_embedding = cr_glove_subset[\'immigration\'],
         pre_trained = cr_glove_subset, N = 5,
         candidates = NULL, norm = "l2")

get_context

Get context words (words within a symmetric window around the tar-
get word/phrase) surrounding a user defined target.

Description

A wrapper function for quanteda’s kwic() function that subsets documents to
where target is present before tokenizing to speed up processing, and concatenates kwic’s pre/post variables into a
column.
Usage

get_context(
  x,
  target,
  window = 6L,
  valuetype = "fixed",
  caseInsensitive = TRUE,
  hard_cut = FALSE,
  what = "word",
  verbose = TRUE
)

Arguments

x (character) vector - this is the set of documents (corpus) of interest.
target (character) vector - these are the target words whose contexts we want to evaluate. This vector may include a single token, a phrase or multiple tokens and/or phrases.
window (numeric) - defines the size of a context (words around the target).
valuetype the type of pattern matching: "glob" for "glob"-style wildcard expressions; "regex" for regular expressions; or "fixed" for exact matching. See valuetype for details.
caseInsensitive logical; if TRUE, ignore case when matching a pattern or dictionary values
hard_cut (logical) - if TRUE then a context must have window x 2 tokens, if FALSE it can have window x 2 or fewer (e.g. if a doc begins with a target word, then context will have window tokens rather than window x 2)
what (character) defines which quanteda tokenizer to use. You will rarely want to change this. For chinese text you may want to set what = "fastestword".
verbose (logical) - if TRUE, report the total number of target instances found.

Value

a data.frame with the following columns:
docname (character) document name to which instances belong to.
target (character) targets.
context (numeric) pre/post variables in kwic() output concatenated.

Note

target in the return data.frame is equivalent to kwic()’s keyword output variable, so it may not match the user-defined target exactly if valuetype is not fixed.
get_cos_sim

Examples

# get context words surrounding the term immigration
context_immigration <- get_context(x = cr_sample_corpus, target = 'immigration',
                                      window = 6, valuetype = "fixed", case_insensitive = FALSE,
                                      hard_cut = FALSE, verbose = FALSE)

get_cos_sim  Given a tokenized corpus, compute the cosine similarities of the resulting ALC embeddings and a defined set of features.

Description

This is a wrapper function for cos_sim() that allows users to go from a tokenized corpus to results with the option to bootstrap cosine similarities and get the corresponding std. errors.

Usage

get_cos_sim(
  x,                     # a (quanteda) tokens-class object
  groups = NULL,        # (numeric, factor, character) a binary variable of the same length as x
  features = character(0), # (character) features of interest
  pre_trained,          # (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
  transform = TRUE,     # (logical) if TRUE (default) apply the 'a la carte' transformation, if FALSE output untransformed averaged embeddings.
  transform_matrix,     # (numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pre-trained embeddings.
  bootstrap = TRUE,     # (logical) if TRUE, use bootstrapping – sample from texts with replacement and re-estimate cosine similarities for each sample. Required to get std. errors. If groups defined, sampling is automatically stratified.
  num_bootstraps = 10,
  as_list = TRUE,
  verbose = TRUE
)

Arguments

x a (quanteda) tokens-class object

get_cos_sim$
get_local_vocab

num_bootstraps (integer) number of bootstraps to use.
as_list (logical) if FALSE all results are combined into a single data.frame If TRUE, a list of data.frames is returned with one data.frame per feature.
verbose (logical) - if TRUE, report the documents that had no overlapping features with the pretrained embeddings provided.

Value

da data.frame or list of data.frames (one for each target) with the following columns:
target (character) rownames of x, the labels of the ALC embeddings.
feature (character) feature terms defined in the features argument.
value (numeric) cosine similarity between x and feature. Average over bootstrapped samples if bootstrap = TRUE.
std.error (numeric) std. error of the similarity value. Column is dropped if bootstrap = FALSE.

Examples

library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts sorrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)

# compute the cosine similarity between each group's embedding
# and a specific set of features
set.seed(2021L)
get_cos_sim(x = immig_toks,
            groups = docvars(immig_toks, 'party'),
            features = c("reform", "enforce"),
            pre_trained = cr_glove_subset,
            transform = TRUE,
            transform_matrix = cr_transform,
            bootstrap = TRUE,
            num_bootstraps = 10,
            as_list = FALSE)

get_local_vocab Identify words common to a collection of texts and a set of pretrained embeddings.

Description

Local vocab consists of the intersect between the set of pretrained embeddings and the collection of texts.
**get_nns**

**Usage**

```
get_local_vocab(context, pre_trained)
```

**Arguments**

- `context` (character) vector of contexts (usually context in `get_context()` output)
- `pre_trained` (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.

**Value**

(character) vector of words common to the texts and pretrained embeddings.

**Examples**

```r
# find local vocab (use it to define the candidate of nearest neighbors)
local_vocab <- get_local_vocab(cr_sample_corpus, pre_trained = cr_glove_subset)
```

**Description**

This is a wrapper function for `nns()` that allows users to go from a tokenized corpus to results with the option to bootstrap cosine similarities and get the corresponding std. errors.

**Usage**

```
get_nns(
  x, 
  N = 10, 
  groups = NULL, 
  candidates = character(0), 
  pre_trained, 
  transform = TRUE, 
  transform_matrix, 
  bootstrap = TRUE, 
  num_bootstraps = 10, 
  as_list = TRUE
)
```
get_nns

Arguments

x a (quanteda) tokens-class object

N (numeric) number of nearest neighbors to return

groups (numeric, factor, character) a binary variable of the same length as x

candidates (character) vector of features to consider as candidates to be nearest neighbor
You may for example want to only consider features that meet a certain count threshold or exclude stop words etc. To do so you can simply identify the set of features you want to consider and supply these as a character vector in the candidates argument.

pre_trained (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.

transform (logical) if TRUE (default) apply the ’a la carte’ transformation, if FALSE output untransformed averaged embeddings.

transform_matrix (numeric) a D x D ’a la carte’ transformation matrix. D = dimensions of pre-trained embeddings.

bootstrap (logical) if TRUE, use bootstrapping – sample from texts with replacement and re-estimate cosine similarities for each sample. Required to get std. errors. If groups defined, sampling is automatically stratified.

num_bootstraps (integer) number of bootstraps to use.

as_list (logical) if FALSE all results are combined into a single data.frame If TRUE, a list of data.frames is returned with one data.frame per target.

Value

a data.frame or list of data.frames (one for each target) with the following columns:

target (character) rownames of x, the labels of the ALC embeddings.

feature (character) features identified as nearest neighbors.

rank (character) rank of feature in terms of similarity with x.

value (numeric) cosine similarity between x and feature. Average over bootstrapped samples if bootstrap = TRUE.

std.error (numeric) std. error of the similarity value. Column is dropped if bootstrap = FALSE.

Examples

library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)
# we limit candidates to features in our corpus
feats <- featnames(dfm(immig_toks))

# compare nearest neighbors between groups
set.seed(2021L)
immig_party_nns <- get_nns(x = immig_toks, N = 10,
groups = docvars(immig_toks, 'party'),
candidates = feats,
pre_trained = cr_glove_subset,
transform = TRUE,
transform_matrix = cr_transform,
bootstrap = TRUE,
um_bootstraps = 10,
as_list = TRUE)

# nearest neighbors of "immigration" for Republican party
immig_party_nns[["R"]]

---

**get_nns_ratio**  
*Given a corpus and a binary grouping variable, computes the ratio of cosine similarities over the union of their respective N nearest neighbors.*

**Description**

This is a wrapper function for nns_ratio() that allows users to go from a tokenized corpus to results with the option to: (1) bootstrap cosine similarity ratios and get the corresponding std. errors. (2) use a permutation test to get empirical p-values for inference.

**Usage**

```r
get_nns_ratio(
  x,
  N = 10,
groups,
numerator = NULL,
candidates = character(0),
pre_trained,
transform = TRUE,
transform_matrix,
bootstrap = TRUE,
um_bootstraps = 10,
permute = TRUE,
um_permutations = 100,
verbose = TRUE
)
```
get_nns_ratio

Arguments

x           a (quanteda) tokens object
N           (numeric) number of nearest neighbors to return. Nearest neighbors consist of
            the union of the top N nearest neighbors of the embeddings in x. If these overlap,
            then resulting N will be smaller than 2*N.
groups      a character or factor variable equal in length to the number of documents
numerator   (character) defines which group is the numerator in the ratio.
candidates  (character) vector of features to consider as candidates to be nearest neighbor
            You may for example want to only consider features that meet a certain count
            threshold or exclude stop words etc. To do so you can simply identify the set
            of features you want to consider and supply these as a character vector in the
            candidates argument.
pre_trained (numeric) a F x D matrix corresponding to pretrained embeddings. F = number
            of features and D = embedding dimensions. rownames(pre_trained) = set of
            features for which there is a pre-trained embedding.
transform   (logical) if TRUE (default) apply the 'a la carte' transformation, if FALSE output
            untransformed averaged embeddings.
transform_matrix (numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pre-
            trained embeddings.
bootstrap   (logical) if TRUE, use bootstrapping – sample from texts with replacement and
            re-estimate cosine similarity ratios for each sample. Required to get std. errors.
            If groups defined, sampling is automatically stratified.
num_bootstraps (integer) number of bootstraps to use.
permute     (logical) if TRUE, compute empirical p-values using permutation test
num_permutations (numeric) number of permutations to use.
verbose     provide information on which group is the numerator

Value

a data.frame with following columns:

feature (character) features in candidates (or all features if candidates not defined), one in-
            stance for each embedding in x.
value (numeric) cosine similarity ratio between x and feature. Average over bootstrapped samples
            if bootstrap = TRUE.
std.error (numeric) std. error of the similarity value. Column is dropped if bootstrap = FALSE.
p.value (numeric) empirical p-value of bootstrapped ratio of cosine similarities if permute =
            TRUE, if FALSE, column is dropped.
group (character) group in groups for which feature belongs to the top N nearest neighbors. If
            "shared", the feature appeared as top nearest neighbor for both groups.
Examples

library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)

# we limit candidates to features in our corpus
feats <- featnames(dfm(immig_toks))

# compute ratio
set.seed(2021L)
immig_nns_ratio <- get_nns_ratio(x = immig_toks,
                                   N = 10,
                                   groups = docvars(immig_toks, 'party'),
                                   numerator = "R",
                                   candidates = feats,
                                   pre_trained = cr_glove_subset,
                                   transform = TRUE,
                                   transform_matrix = cr_transform,
                                   bootstrap = TRUE,
                                   num_bootstraps = 5,
                                   permute = TRUE,
                                   num_permutations = 5,
                                   verbose = FALSE)

head(immig_nns_ratio)

---

get_seq_cos_sim

Calculate cosine similarities between target word and candidates words over sequenced variable using ALC embedding approach

Description

Calculate cosine similarities between target word and candidates words over sequenced variable using ALC embedding approach

Usage

get_seq_cos_sim(
  x,
  seqvar,
  target,
  candidates,
  pre_trained,
  transform_matrix,
  num_bootstraps = 5,
  permute = TRUE,
  num_permutations = 5,
  bootstrap = TRUE,
  verbose = FALSE
)
get_seq_cos_sim

window = 6,
valuetype = "fixed",
case_insensitive = TRUE,
hard_cut = FALSE,
verbose = TRUE
)

Arguments

x (character) vector - this is the set of documents (corpus) of interest
seqvar ordered variable such as list of dates or ordered ideology scores
target (character) vector - target word
candidates (character) vector of features of interest
pre_trained (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
transform_matrix (numeric) a D x D ‘a la carte’ transformation matrix. D = dimensions of pre-trained embeddings.
window (numeric) - defines the size of a context (words around the target).
valuetype the type of pattern matching: "glob" for "glob"-style wildcard expressions; "regex" for regular expressions; or "fixed" for exact matching. See valuetye for details.
case_insensitive logical; if TRUE, ignore case when matching a pattern or dictionary values
hard_cut (logical) - if TRUE then a context must have window x 2 tokens, if FALSE it can have window x 2 or fewer (e.g. if a doc begins with a target word, then context will have window tokens rather than window x 2)
verbose (logical) - if TRUE, report the total number of target instances found.

Value

a data.frame with one column for each candidate term with corresponding cosine similarity values and one column for seqvar.

Examples

library(quanteda)

# gen sequence var (here: year)
docvars(cr_sample_corpus, 'year') <- rep(2011:2014, each = 50)
cos_simsdf <- get_seq_cos_sim(x = cr_sample_corpus,
seqvar = docvars(cr_sample_corpus, 'year'),
target = "equal",
candidates = c("immigration", "immigrants"),
pre_trained = cr_glove_subset,
transform_matrix = cr_transform)
Given a set of embeddings and a set of candidate neighbors, find the top N nearest neighbors.

**Usage**

```
nns(x, N = 10, candidates = character(0), pre_trained, as_list = TRUE)
```

**Arguments**

- `x`: a (quanteda) dem-class or fem-class object.
- `N`: (numeric) number of nearest neighbors to return
- `candidates`: (character) vector of features to consider as candidates to be nearest neighbor. You may for example want to only consider features that meet a certain count threshold or exclude stop words etc. To do so you can simply identify the set of features you want to consider and supply these as a character vector in the candidates argument.
- `pre_trained`: (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
- `as_list`: (logical) if FALSE all results are combined into a single data.frame If TRUE, a list of data.frames is returned with one data.frame per target.

**Value**

A data.frame or list of data.frames (one for each target) with the following columns:

- `target`: (character) rownames of x, the labels of the ALC embeddings.
- `feature`: (character) features identified as nearest neighbors.
- `rank`: (character) rank of feature in terms of similarity with x.
- `value`: (numeric) cosine similarity between x and feature.

**Examples**

```
library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)
```
# build document-feature matrix
immig_dfm <- dfm(immig_toks)

# construct document-embedding-matrix
immig_dem <- dem(immig_dfm, pre_trained = cr_glove_subset,
transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)

# to get group-specific embeddings, average within party
immig_wv_party <- dem_group(immig_dem, groups = immig_dem@docvars$party)

# find nearest neighbors by party
# setting as_list = FALSE combines each group's
# results into a single tibble (useful for joint plotting)
immig_nns <- nns(immig_wv_party, pre_trained = cr_glove_subset,
N = 5, candidates = immig_wv_party@features, as_list = TRUE)

nns_ratio

Computes the ratio of cosine similarities for two embeddings over the union of their respective top N nearest neighbors.

Description
Computes the ratio of cosine similarities between group embeddings and features—that is, for any given feature it first computes the similarity between that feature and each group embedding, and then takes the ratio of these two similarities. This ratio captures how "discriminant" a feature is of a given group. Values larger (smaller) than 1 mean the feature is more (less) discriminant of the group in the numerator (denominator).

Usage
nns_ratio(
  x,
  N = 10,
  numerator = NULL,
  candidates = character(0),
  pre_trained,
  verbose = TRUE
)

Arguments
x a (quanteda) dem-class or fem-class object.
N (numeric) number of nearest neighbors to return. Nearest neighbors consist of the union of the top N nearest neighbors of the embeddings in x. If these overlap, then resulting N will be smaller than 2*N.
numerator (character) defines which group is the numerator in the ratio
candidates (character) vector of features to consider as candidates to be nearest neighbor. You may for example want to only consider features that meet a certain count threshold or exclude stop words etc. To do so you can simply identify the set of features you want to consider and supply these as a character vector in the candidates argument.

pre_trained (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.

verbose report which group is the numerator and which group is the denominator.

Value

a data.frame with following columns:

feature (character) features in candidates (or all features if candidates not defined), one instance for each embedding in x.

value (numeric) ratio of cosine similarities.

Examples

library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr\*", window = 6L)

# build document-feature matrix
immig_dfm <- dfm(immig_toks)

# construct document-embedding-matrix
immig_dem <- dem(immig_dfm, pre_trained = cr_glove_subset, transform = TRUE, transform_matrix = cr_transform, verbose = FALSE)

# to get group-specific embeddings, average within party
immig_wv_party <- dem_group(immig_dem, groups = immig_dem@docvars$party)

# compute the cosine similarity between each party's embedding and a specific set of features
nns_ratio(x = immig_wv_party, N = 10, numerator = "R", candidates = immig_wv_party@features, pre_trained = cr_glove_subset, verbose = FALSE)
**permute_contrast**

Permute similarity and ratio computations

**Description**

Permute similarity and ratio computations

**Usage**

```r
permute_contrast(
  target_embeddings1 = NULL,
  target_embeddings2 = NULL,
  pre_trained = NULL,
  candidates = NULL,
  norm = NULL
)
```

**Arguments**

- `target_embeddings1`: ALC embeddings for group 1
- `target_embeddings2`: ALC embeddings for group 2
- `pre_trained`: a V x D matrix of numeric values - pretrained embeddings with V = size of vocabulary and D = embedding dimensions
- `candidates`: character vector defining the candidates for nearest neighbors - e.g. output from `get_local_vocab`
- `norm`: character = c("l2", "none") - set to 'l2' for cosine similarity and to 'none' for inner product (see `?sim2` in `text2vec`)

**Value**

a list with three elements, nns for group 1, nns for group 2 and nns_ratio comparing with ratios of similarities between the two groups

**permute_ols**

Permute OLS

**Description**

Estimate empirical p-value using permutated regression

**Usage**

```r
permute_ols(Y = NULL, X = NULL)
```
Arguments

\(Y\) vector of regression model’s dependent variable (embedded context)

\(X\) data.frame of model independent variables (covariates)

Value

list with two elements, \(\text{betas} = \) list of beta_coefficients (D dimensional vectors); \(\text{normed_betas} = \) tibble with the norm of the non-intercept coefficients

Description

A way of visualizing the top nearest neighbors of a pair of ALC embeddings that captures how “discriminant” each feature is of each embedding (group).

Usage

\[
\text{plot_nns_ratio}(x, \alpha = 0.01, \text{horizontal} = \text{TRUE})
\]

Arguments

\(x\) output of \(\text{get_nns_ratio}\)

\(\alpha\) (numerical) between 0 and 1. Significance threshold to identify significant values. These are denoted by a * on the plot.

\(\text{horizontal}\) (logical) defines the type of plot. if \(\text{TRUE}\) results are plotted on 1 dimension. If \(\text{FALSE}\), results are plotted on 2 dimensions, with the second dimension capturing the ranking of cosine ratio similarities.

Value

a \texttt{ggplot-class} object.

Examples

```r
library(ggplot2)
library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)

# we limit candidates to features in our corpus
```
prototypical_context <- featnames(dfm(immig_toks))

# compute ratio
set.seed(2021L)
immig_nns_ratio <- get_nns_ratio(x = immig_toks,
                                  N = 10,
                                  groups = docvars(immig_toks, 'party'),
                                  numerator = "R",
                                  candidates = feats,
                                  pre_trained = cr_glove_subset,
                                  transform = TRUE,
                                  transform_matrix = cr_transform,
                                  bootstrap = TRUE,
                                  num_bootstraps = 10,
                                  permute = TRUE,
                                  num_permutations = 10,
                                  verbose = FALSE)

plot_nns_ratio(x = immig_nns_ratio, alpha = 0.01, horizontal = TRUE)

prototypical_context  Find most "prototypical" contexts.

Description

Contexts most similar on average to the full set of contexts.

Usage

prototypical_context(
  context,
  pre_trained,
  transform = TRUE,
  transform_matrix,
  N = 3,
  norm = "l2"
)

Arguments

context  (character) vector of texts - context variable in get_context output
pre_trained  (numeric) a F x D matrix corresponding to pretrained embeddings. F = number of features and D = embedding dimensions. rownames(pre_trained) = set of features for which there is a pre-trained embedding.
transform  (logical) - if TRUE (default) apply the a la carte transformation, if FALSE ouput untransformed averaged embedding.
transform_matrix  (numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pre-trained embeddings.
run_ols

\[ N \] (numeric) number of most "prototypical" contexts to return.

\[ \text{norm} \] (character) - how to compute similarity (see \texttt{?text2vec::sim2}):

"l2" cosine similarity

"none" inner product

Value

a \texttt{data.frame} with the following columns:

- \texttt{doc_id} (integer) document id.
- \texttt{typicality_score} (numeric) average similarity score to all other contexts
- \texttt{context} (character) contexts

Examples

```r
# find contexts of immigration
context_immigration <- get_context(x = cr_sample_corpus, target = 'immigration',
                                      window = 6, valuetype = "fixed", case_insensitive = TRUE,
                                      hard_cut = FALSE, verbose = FALSE)

# identify top N prototypical contexts and compute typicality score
pt_context <- prototypical_context(context = context_immigration$context,
                                     pre_trained = cr_glove_subset,
                                     transform = TRUE,
                                     transform_matrix = cr_transform,
                                     N = 3, norm = 'l2')
```

run_ols  \hspace{1cm} Run OLS

Description

Bootstrap model coefficients and standard errors

Usage

```r
run_ols(Y = NULL, X = NULL)
```

Arguments

- \texttt{Y} vector of regression model’s dependent variable (embedded context)
- \texttt{X} \texttt{data.frame} of model independent variables (covariates)

Value

list with two elements, \texttt{betas} = list of beta_coefficients (D dimensional vectors); \texttt{normed_betas} = tibble with the norm of the non-intercept coefficients
tokens_context  

Get the tokens of contexts surrounding user defined patterns

Description

This function uses quanteda’s kwic() function to find the contexts around user defined patterns (i.e. target words/phrases) and return a tokens object with the tokenized contexts and corresponding document variables.

Usage

tokens_context(
  x,
  pattern,
  window = 6L,
  valuetype = c("glob", "regex", "fixed"),
  case_insensitive = TRUE,
  hard_cut = FALSE,
  verbose = TRUE
)

Arguments

x            a (quanteda) tokens-class object
pattern      a character vector, list of character vectors, dictionary, or collocations object. See pattern for details.
window       the number of context words to be displayed around the keyword
valuetype    the type of pattern matching: "glob" for "glob"-style wildcard expressions; "regex" for regular expressions; or "fixed" for exact matching. See valuetype for details.
case_insensitive logical; if TRUE, ignore case when matching a pattern or dictionary values
hard_cut     (logical) - if TRUE then a context must have window x 2 tokens, if FALSE it can have window x 2 or fewer (e.g. if a doc begins with a target word, then context will have window tokens rather than window x 2)
verbose      (logical) if TRUE, report the total number of instances per pattern found

Value

a (quanteda) tokens-class. Each document in the output tokens object inherits the document variables (docvars) of the document from whence it came, along with a column registering corresponding the pattern used. This information can be retrieved using docvars().
Examples

```r
library(quanteda)

# tokenize corpus
toks <- tokens(cr_sample_corpus)

# build a tokenized corpus of contexts surrounding a target term
immig_toks <- tokens_context(x = toks, pattern = "immigr*", window = 6L)
```
Index

* bootstrap_nns
  bootstrap_nns, 3
* compute_transform
  compute_transform, 10
* conText
  conText, 11
* contrast_nns
  contrast_nns, 13
* cos_sim
  cos_sim, 15
* datasets
  cr_glove_subset, 16
  cr_sample_corpus, 17
  cr_transform, 17
* dem_group
  dem_group, 19
* dem
  dem, 18
* embed_target
  embed_target, 20
* feature_sim
  feature_sim, 21
* fem
  fem, 22
* find_nns
  find_nns, 24
* get_context
  get_context, 24
* get_cos_sim
  get_cos_sim, 26
* get_local_vocab
  get_local_vocab, 27
* get_nns_ratio
  get_nns_ratio, 30
* get_nns
  get_nns, 28
* nns_ratio
  nns_ratio, 35
* nns
  nns, 34
* plot_nns_ratio
  plot_nns_ratio, 38
* tokens_context
  tokens_context, 41
  bootstrap_contrast, 3
  bootstrap_nns, 3
  bootstrap_ols, 5
  bootstrap_similarity, 6
  build_conText, 6
  build_dem, 7
  build_fem, 8
  compute_contrast, 8
  compute_similarity, 9
  compute_transform, 10
  conText, 11
  contrast_nns, 13
  cos_sim, 15
  cr_glove_subset, 16
  cr_sample_corpus, 17
  cr_transform, 17
  dem, 18
  dem_group, 19
  dictionary, 12, 25, 33, 41
  embed_target, 20
  feature_sim, 21
  fem, 22
  find_cos_sim, 23
  find_nns, 24
  get_context, 24
  get_cos_sim, 26
  get_local_vocab, 27
  get_nns, 28
  get_nns_ratio, 30
  get_seq_cos_sim, 32
nns, 34
nns_ratio, 35

pattern, 41
permute_contrast, 37
permute_ols, 37
plot_nns_ratio, 38
prototypical_context, 39

run_ols, 40

tokens_context, 41

valuetype, 12, 25, 33, 41