Package ‘conText’

April 5, 2022

Version 1.1.0

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

Description A fast, flexible and transparent framework to estimate context-specific word and short document embeddings using the 'a la carte' embeddings approach developed by Khodak et al. (2018) <arXiv:1805.05388> and evaluate hypotheses about covariate effects on embeddings using the regression framework developed by Rodriguez et al. (2021)<https://github.com/prodriguezsosa/EmbeddingRegression>.

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), tidyr (>= 1.1.3), 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 SnowballC (>= 0.7.0), hunspell, knitr, rmarkdown, formatR

NeedsCompilation no

Author Pedro L. Rodriguez [aut, cre, cph]
(<https://orcid.org/0000-0003-4762-4550>),
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 2022-04-05 07:10:10 UTC

R topics documented:

`bootstrap_contrast` *Bootstrap similarity and ratio computations*

Description

Bootstrap similarity and ratio computations

Usage

```
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` *Bootstrap nearest neighbors*

Description

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

Usage

```
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 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 pretrained 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
context_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(context_immigration$context, pre_trained = cr_glove_subset)

set.seed(42L)
nns_immigration <- bootstrap_nns(context = context_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 *Boostrap similarity vector*

Description

Boostrap similarity vector

Usage

```
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

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

Arguments

`Class` defines the class of this object (fixed)

`x_conText` a dgCMatrix class matrix

`normed_cofficients` a data.frame withe the normed coefficients and other statistics

`features` features used in computing the embeddings

`Dimnames` row (features) and columns (NULL) names

`build_dem` *build a* `dem-class` *object*

Description

build a `dem-class` object

Usage

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

Arguments

`Class` defines tha 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
)
```

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

`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

```
conText(
  formula,
  data,
  pre_trained,
  transform = TRUE,
  transform_matrix,
  bootstrap = TRUE,
  num_bootstraps = 100,
  stratify = FALSE,
  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 in 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`. If you wish to use all covariates on the RHS use `immigrant ~ .`. Any `character` or `factor` covariates will automatically be converted to a set of binary (`0`/`1`s) indicator variables for each group, leaving the first level out of the regression.

`data` a quanteda `tokens-class` object with the necessary document variables. Covariates must be either binary indicator variables or "trasnformable" 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 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 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 discrete 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 `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 documents that had no overlapping features with the pretrained embeddings provided.

Value

a `conText-class` object - a D x 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)

## 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 = FALSE,
                  permute = TRUE, num_permutations = 100,
                  window = 6, case_insensitive = TRUE,
                  verbose = FALSE)

# notice, character/factor 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_cofficients
```

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,
  num_bootstraps = 20,
  permute = TRUE,
  num_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 pretrained 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.

`num_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

a 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 similarties. Column is dropped if bootsrap = FALSE.

`p.value` (numeric) empirical p-value. Column is dropped if permute = FALSE.

Examples

```
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

```
cos_sim(x, pre_trained, features = NULL, stem = FALSE, 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.
stem	(logical) - If TRUE, both features and rownames(pre_trained) are stemmed and average cosine similarities are reported. We recommend you remove misspelled words from pre_trained as these can significantly influence the average.
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. NA if is.null(rownames(x)).
- feature (character) feature terms defined in the features argument.
- value (numeric) cosine similarity between x and feature.

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)

# 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(x = 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 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

https://www.dropbox.com/s/p84wzv8bdmziog8/cr_glove.R?dl=0

`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 For nominate scores see: https://voteview.com/data

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

nominate_dim1 dimension 1 of the nominate score ...

Source

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

https://www.dropbox.com/s/p84wzv8bdmziog8/cr_glove.R?dl=0

`dem` *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 ouput 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

```
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)

# 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 sorrounding 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,
  pre_trained,
  transform = TRUE,
  transform_matrix,
  aggregate = TRUE,
  verbose = TRUE
)
```

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 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

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

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

feature_sim — *Given two feature-embedding-matrices, compute "parallel" cosine similarities between overlapping features.*

Description

Efficient way of comparing two corpora along many features simultaneously.

Usage

```
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

```
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 featureco-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

```
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 ouput 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 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 ?sim2)

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”,
  stem = FALSE
)
```

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

`stem` (logical) - whether to stem candidates when evaluating nns. Default is FALSE. If TRUE, candidate stems are ranked by their average cosine similarity to the target. We recommend you remove misspelled words from candidate set `candidates` as these can significantly influence the average.

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", stem = FALSE)
```

---

get_context — *Get context words (words within a symmetric window around the target word/phrase) sorrounding 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 `context` column.

Usage

```
get_context(
  x,
  target,
  window = 6L,
  valuetype = "fixed",
  case_insensitive = 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.

`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)
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.

Examples

```
# get context words sorrounding 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,
  groups = NULL,
  features = character(0),
  pre_trained,
  transform = TRUE,
  transform_matrix,
  bootstrap = TRUE,
  num_bootstraps = 10,
  stem = FALSE,
  as_list = TRUE
)
```

Arguments

`x` a (quanteda) `tokens-class` object

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

`features` (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` (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 pretrained 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.

`stem` (logical) - If TRUE, both `features` and `rownames(pre_trained)` are stemmed and average cosine similarities are reported. We recommend you remove misspelled words from `pre_trained` as these can significantly influence the average.

`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. 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,
            stem = TRUE,
            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.

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

```
# 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)
```

get_ncs *Given a set of tokenized contexts, find the top N nearest contexts.*

Description

This is a wrapper function for `ncs()` 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_ncs(
  x,
  N = 5,
  groups = NULL,
  pre_trained,
  transform = TRUE,
  transform_matrix,
  bootstrap = TRUE,
  num_bootstraps = 10,
  as_list = TRUE
)
```

Arguments

`x` a (quanteda) `tokens-class` object

`N` (numeric) number of nearest contexts to return

`groups` a character or factor variable equal in length to the number of documents

`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 pretrained embeddings.

`bootstrap` (logical) if TRUE, use bootstrapping – sample from `x` 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 embedding

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. `NA` if `is.null(rownames(x))`.

`context` (character) contexts collapsed into single documents (i.e. untokenized).

`rank` (character) rank of context in terms of similarity with `x`.

`value` (numeric) cosine similarity between `x` and context.

`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, rm_keyword = FALSE)

# compare nearest contexts between groups
set.seed(2021L)
immig_party_ncs <- get_ncs(x = immig_toks,
                           N = 10,
                           groups = docvars(immig_toks, 'party'),
                           pre_trained = cr_glove_subset,
                           transform = TRUE,
                           transform_matrix = cr_transform,
                           bootstrap = TRUE,
                           num_bootstraps = 10,
                           as_list = TRUE)

# nearest neighbors of "immigration" for Republican party
immig_party_ncs[["D"]]
```

`get_nns` *Given a tokenized corpus and a set of candidate neighbors, find the top N nearest neighbors.*

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,
  stem = FALSE,
  as_list = TRUE
)
```

Arguments

`x` a (quanteda) `tokens-class` object

`N` (numeric) number of nearest neighbors to return

`groups` a character or factor variable equal in length to the number of documents

`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 ouput untransformed averaged embeddings.

`transform_matrix` (numeric) a D x D 'a la carte' transformation matrix. D = dimensions of pretrained embeddings.

`bootstrap` (logical) if TRUE, use bootstrapping – sample from `x` with replacement and reestimate 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.

`stem` (logical) - whether to stem candidates when evaluating nns. Default is FALSE. If TRUE, candidate stems are ranked by their average cosine similarity to the target. We recommend you remove misspelled words from candidate set `candidates` as these can significantly influence the average.

`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 group.

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. NA if is.null(rownames(x)).

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 sorrounding 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,
                           num_bootstraps = 10,
                           stem = TRUE,
                           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

```
get_nns_ratio(
  x,
  N = 10,
  groups,
  numerator = NULL,
  candidates = character(0),
  pre_trained,
  transform = TRUE,
  transform_matrix,
  bootstrap = TRUE,
  num_bootstraps = 10,
  permute = TRUE,
  num_permutations = 100,
  stem = FALSE,
  verbose = TRUE
)
```

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 nuemerator 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 certian 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 ouput untransformed averaged embeddings.

`transform_matrix`

(numeric) a D x D ’a la carte’ transformation matrix. D = dimensions of pretrained 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.

stem (logical) - whether to stem candidates when evaluating nns. Default is FALSE. If TRUE, candidate stems are ranked by their average cosine similarity to the target. We recommend you remove misspelled words from candidate set `candidates` as these can significantly influence the average.

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 instance 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 sorrounding 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,
  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 iseology 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 pretrained 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 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 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)
```

ncs *Given a set of embeddings and a set of tokenized contexts, find the top N nearest contexts.*

Description

Given a set of embeddings and a set of tokenized contexts, find the top N nearest contexts.

Usage

```
ncs(x, contexts_dem, contexts = NULL, N = 5, as_list = TRUE)
```

Arguments

x	a (quanteda) dem-class or fem-class object.
contexts_dem	a dem-class object corresponding to the ALC embeddings of candidate contexts.
contexts	a (quanteda) tokens-class object of tokenized candidate contexts. Note, these must correspond to the same contexts in contexts_dem. If NULL, then the context (document) ids will be output instead of the text.
N	(numeric) number of nearest contexts to return
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 embedding

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. `NA` if `is.null(rownames(x))`.

`context` (character) contexts collapsed into single documents (i.e. untokenized). If `contexts` is NULL then this variable will show the context (document) ids which you can use to merge.

`rank` (character) rank of context in terms of similarity with `x`.

`value` (numeric) cosine similarity between `x` and context.

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, rm_keyword = FALSE)

# 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 contexts by party
# setting as_list = FALSE combines each group's
# results into a single data.frame (useful for joint plotting)
ncs(x = immig_wv_party, contexts_dem = immig_dem,
contexts = immig_toks, N = 5, as_list = TRUE)
```

`nns` *Given a set of embeddings and a set of candidate neighbors, find the top N nearest neighbors.*

Description

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,
  stem = FALSE,
  as_list = TRUE
)
```

Arguments

`x`	a `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.
`stem`	(logical) - whether to stem candidates when evaluating nns. Default is FALSE. If TRUE, candidate stems are ranked by their average cosine similarity to the target. We recommend you remove misspelled words from candidate set `candidates` as these can significantly influence the average.
`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 group.

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. `NA` if `is.null(rownames(x))`.

`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 sorrounding 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, stem = TRUE, 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,
  stem = FALSE,
  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 nuemerator 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 certian 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.

`stem` (logical) - whether to stem candidates when evaluating nns. Default is FALSE. If TRUE, candidate stems are ranked by their average cosine similarity to the target. We recommend you remove misspelled words from candidate set `candidates` as these can significantly influence the average.

`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 sorrounding 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)
```

```
# with stemming
nns_ratio(x = immig_wv_party, N = 10, numerator = "R",
candidates = immig_wv_party@features,
pre_trained = cr_glove_subset, stem = TRUE, verbose = FALSE)
```

permute_contrast *Permute similarity and ratio computations*

Description

Permute similarity and ratio computations

Usage

```
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

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

plot_nns_ratio *Plot output of* `get_nns_ratio()`

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

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

Arguments

x	output of get_nns_ratio
alpha	(numerical) betwee 0 and 1. Significance threshold to identify significant values. These are denoted by a * on the plot.
horizontal	(logical) defines the type of plot. if TRUE results are plotted on 1 dimension. If FALSE, results are plotted on 2 dimensions, with the second dimension catpuring the ranking of cosine ratio similarties.

Value

a `ggplot-class` object.

Examples

```
library(ggplot2)
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)

# 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 = 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 pretrained embeddings.

`N` (numeric) number of most "prototypical" contexts to return.

`norm` (character) - how to compute similarity (see ?text2vec::sim2):

`"l2"` cosine similarity

`"none"` inner product

Value

a `data.frame` with the following columns:

`doc_id` (integer) document id.

`typicality_score` (numeric) average similarity score to all other contexts

`context` (character) contexts

Examples

```
# 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 *Run OLS*

Description

Bootstrap model coefficients and standard errors

Usage

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

`tokens_context` *Get the tokens of contexts sorrounding 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,
  rm_keyword = TRUE,
  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)

`rm_keyword` (logical) if FALSE, keyword matching pattern is included in the tokenized contexts

`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

```
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)
```

Index