Package ‘preText’

January 12, 2018

Type Package

Title Diagnostics to Assess the Effects of Text Preprocessing Decisions

Version 0.6.2

Date 2018-01-12

Author Matthew J. Denny <mdenny@psu.edu>, Arthur Spirling <as9934@nyu.edu>,

Maintainer Matthew J. Denny <mdenny@psu.edu>

Description Functions to assess the effects of different text preprocessing decisions on the inferences drawn from the resulting document-term matrices they generate.

License GPL-3

Imports quanteda, ggplot2, vegan, grid, parallel, topicmodels, cowplot, ecodist, proxy, reshape2

Depends R (>= 3.3.0)

RoxygenNote 6.0.1

LazyData TRUE

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-12 17:11:53 UTC

R topics documented:

calculate_prediction_errors ................................................. 2
dfm_scaling_test ............................................................. 3
document_position_plots ....................................................... 4
factorial_preprocessing ...................................................... 5
mantel_comparison ............................................................ 7
mantel_comparison_to_base ................................................... 8
calculate_prediction_errors

Description
Use scaled positions to predict preprocessing decisions.

Usage
calculate_prediction_errors(positions_list, preprocessing_choices)

Arguments
positions_list  A list of scaled document positions generated by the `scaling_comparison()` function and returned from that function in the `$scaled_positions` slot in the list object.
preprocessing_choices  A data frame containing binary indicators of whether each preprocessing decision was applied for each dfm. This is returned by the `factorial_preprocessing()` function as part of its output.

Value
A vector of mean prediction errors.
**dfm_scaling_test**

*Comparison of dfms using N-dimensional scaling, with a test for difference from the mean dfm scaled position.*

### Examples

```r
## Not run:
# *** This function is used automatically inside of the pretext() function.
# load the package
library(pretext)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
# scale documents
scaling_results <- scaling_comparison(preprocessed_documents$dfm_list,
  dimensions = 2,
  distance_method = "cosine",
  verbose = TRUE)

# get prediction errors
pred_errors <- calculate_prediction_errors(
  scaling_results$scaled_positions,
  preprocessed_documents$choices)

## End(Not run)
```

### Description

Scale each dfm into a N-d space and test for outliers.

### Usage

```r
dfm_scaling_test(scaling_results, labels, dimensions = 2,
  distance_method = "cosine", method = c("distances", "positions"),
  return_positions = FALSE)
```

### Arguments

- **scaling_results**
  A list object produced by the `scaling_comparison()` function.
- **labels**
  A character vector with labels for each dfm. This can be extracted from the `$labels` field of the output from the `factorial_preprocessing()` function.
- **dimensions**
  The number of dimensions to be used by the multidimensional scaling algorithm. Defaults to 2.
distance_method
The method that should be used for calculating distances between dfms. Defaults to "cosine".

method
Should the raw distances or scaled document positions be used for scaling? Can be one of c("distances", "positions"), defaults to "distances".

return_positions
Logical indicating whether dfm positions should be returned as a data.frame. Defaults to FALSE.

Value
A result list object, or a plot, or both.

Examples
```r
## Not run:
# *** This function is used automatically inside of the pretext() function.
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
# scale documents
scaling_results <- scaling_comparison(preprocessed_documents$dfm_list,
  dimensions = 2,
  distance_method = "cosine",
  verbose = TRUE)
# now perform the scaling test
dfm_scaling_test(scaling_results,
  labels = preprocessed_documents$labels)
```

---

document_position_plots

**Document Position Plots**

**Description**
Plot Procrustes transformed scaled positions of each document under each preprocessing regime.

**Usage**
```r
document_position_plots(positions_list, num_cols = 10, colors = NULL,
  decision_colors = NULL)
```
factorial_preprocessing

Arguments

positions_list A list of scaled document positions generated by the ‘scaling_comparison()’ and returned in the ‘$scaled_positions’ field.

num_cols The number of columns to use in combining plots into a large tiled plot.

colors Optional vector of document colors to distinguish groups.

decision_colors Defaults to NULL, if desired, the user should provide a vector of logical values of length equal to the number of preprocessing decisions. Can be used to bifurcate the points within a single plot to show the effects of a particular decision. Points in the TRUE class will be colors BLUE and those in the FALSE class will be colored red.

Value

A list of ggplot2 objects.

Examples

## Not run:
# *** This function is used automatically inside of the preText() function.
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
# scale documents
scaling_results <- scaling_comparison(preprocessed_documents$dfm_list,
  dimensions = 2,
  distance_method = "cosine",
  verbose = TRUE)
# plot scaled positions
document_position_plots(scaling_results$scaled_positions,
  num_cols = 10,
  colors = NULL,
  decision_colors = NULL)

## End(Not run)
factorial_preprocessing

Description

Preprocesses a corpus of texts into a document-frequency matrix in 128 different ways.

Usage

factorial_preprocessing(text, use_ngrams = TRUE, infrequent_term_threshold = 0.01, parallel = FALSE, cores = 1, intermediate_directory = NULL, parameterization_range = NULL, return_results = TRUE, verbose = TRUE)

Arguments

text A vector of strings (one per document) or quanteda corpus object from which we wish to form a document-term matrix.

use_ngrams Option to extract 1, 2, and 3-grams from the text as another potential preprocessing step. Defaults to TRUE.

infrequent_term_threshold A proportion threshold at which infrequent terms are to be filtered. Defaults to 0.01 (terms that appear in less than 1 percent of documents).

parallel Logical indicating whether factorial preprocessing should be performed in parallel. Defaults to FALSE.

cores Defaults to 1, can be set to any number less than or equal to the number of cores on one’s computer.

intermediate_directory Optional path to a directory where each dfm will be saved as an intermediate step. The file names will follow the convention intermediate_dfm_i.Rdata, where i is the index of the combination of preprocessing choices. The function will then attempt to read all of the dfm’s back into a list if return_results = TRUE (by default), or simply end the function call if return_results = FALSE. This can be a useful option if the user is preprocessing a corpus that would make a dfm list that was impractical to work with due to its size.

parameterization_range Defaults to NULL, but can be set to a numeric vector of indexes relating to preprocessing decisions. This can be used to restart large analyses after power failure.

return_results Defaults to TRUE, can be set to FALSE to prevent an overly large dfm list from being created.

verbose Logical indicating whether more information should be printed to the screen to let the user know about progress in preprocessing. Defaults to TRUE.

Value

A list object containing permutations of the document-term matrix.
mantel_comparison

Examples

```r
## Not run:
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)

## End(Not run)
```

---

mantel_comparison Ensemble Mantel Tests

Description

Calculates Mantel test statistics for differences between distance matrices for a list of distance matrices (one per preprocessing method) supplied by the `scaling_comparison()` function.

Usage

```r
mantel_comparison(distance_matrices, labels = NULL, permutations = 1000)
```

Arguments

- **distance_matrices**
  A list of document distance matrices generated by the `scaling_comparison()` and returned in the `$distance_matrices` field.

- **labels**
  Optional argument giving names for each preprocessing step. This is generated by the `factorial_preprocessing()` function and returned in the `$labels` field.

- **permutations**
  The number of permutations to be used in each Mantel test. Defaults to 1000.

Value

A result list object where the first entry is a matrix summarizing mantel test statistics. The second object in the list is a matrix of the values described above. The third object is a list of all raw mantel results.
Examples

```r
## Not run:
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
# scale documents
scaling_results <- scaling_comparison(preprocessed_documents$dfm_list,
  dimensions = 2,
  distance_method = "cosine",
  verbose = TRUE)

## run mantel tests
mantel_results <- mantel_comparison(scaling_results$distance_matrices,
  labels = preprocessed_documents$labels,
  permutations = 1000)
```

## End(Not run)

---

**mantel_comparison_to_base**

*Ensemble Mantel Tests*

**Description**

Calculates Mantel test statistics for differences between distance matrices for a list of distance matrices (one per preprocessing method) supplied by the `scaling_comparison()` function to a base case – (usually the no-preprocessing specification).

**Usage**

```r
mantel_comparison_to_base(distance_matrices, names = NULL,
  permutations = 1000, base_dfm_index = 128, text_size = 1,
  return_values = FALSE)
```

**Arguments**

- `distance_matrices`  
  A list of document distance matrices from the `$distance_matrices` field of the output from the `scaling_comparison()` function.

- `names`  
  Optional argument giving names for each preprocessing step.

- `permutations`  
  The number of permutations to be used in each Mantel test. Defaults to 1000.
**optimal_k_comparison**

base_dfm_index Which dfm should be used as a base case for comparing r statistics with bootstrapped confidence intervals.

text_size The ‘cex’ for the x-labels, defaults to 1.

return_values Logical indicating whether test statistics and confidence bounds should be returned as a data.frame or not. Defaults to FALSE.

**Value**

A data.frame with mantel statistics and 95 percent confidence intervals comparing all other preprocessing choices to base case, and/or a plot of confidence intervals.

**Examples**

```r
## Not run:
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
# scale documents
scaling_results <- scaling_comparison(preprocessed_documents$dfm_list,
  dimensions = 2,
  distance_method = "cosine",
  verbose = TRUE)
# run mantel comparison to base and plot
mantel_comparison_to_base(scaling_results$distance_matrices,
  names = preprocessed_documents$labels,
  permutations = 1000)
## End(Not run)
```

---

**optimal_k_comparison  Optimal Topic Model k Comparison**

**Description**

Calculate the optimal number of topics for LDA using perplexity for each dfm.

**Usage**

```r
optimal_k_comparison(cross_validation_train_document_indices,
cross_validation_test_document_indices, dfm_object_list = NULL,
topics = c(2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100), names = NULL,
parallel = FALSE, cores = 1, intermediate_file_directory = NULL,
intermediate_file_names = NULL)
```
Arguments

cross_validation_train_document_indices
A list of numeric vectors where the length of the list is equal to the number
of splits to be used for cross validation, and each vector contains the numeric
indices of documents to be used for training.

cross_validation_test_document_indices
A list of numeric vectors where the length of the list is equal to the number
of splits to be used for cross validation, and each vector contains the numeric
indices of documents to be used for testing.

dfm_object_list
An optional list of quanteda dfm() objects. If none are provided, then interme-
diate files will be used.

topics
A numeric vector containing the numbers of topics to search over. Defaults to
`c(2,5,10,20,30,40,50,60,70,80,90,100)`.

names
optional names for each dfm to make downstream interpretation easier. Defaults
to NULL.

parallel
Logical indicating whether model fitting should be performed in parallel. De-
defaults to FALSE.

cores
Defaults to 1, can be set to any number less than or equal to the number of cores
on one’s computer.

intermediate_file_directory
Optional directory containing Rdata files for each of the factorial preprocessing
combinations.

intermediate_file_names
Optional vector of file names for intermediate Rdata files – one per combination.

Value

A vector containing the optimal k for each dfm.

Examples

```r
## Not run:
set.seed(12345)
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
cross_validation_splits <- 10
# create 10 test/train splits
train_inds <- vector(mode = "list", length = cross_validation_splits)
test_inds <- vector(mode = "list", length = cross_validation_splits)
```
```r
# sample CV indices
for (i in 1:cross_validation_splits) {
  test <- sample(1:length(uk_Manifestos),
                  size = round(length(uk_Manifestos)/5),
                  replace = FALSE)
  train <- 1:length(uk_Manifestos)
  for (j in 1:length(test)) {
    train <- train[-which(train == test[j])]
  }
  train_inds[[i]] <- train
  test_inds[[i]] <- test
}
# get the optimal number of topics (this will take a very long time):
optimal_k <- optimal_k_comparison(
  train_inds,
  test_inds,
  preprocessed_documents$dfm_list,
  topics = c(25,50,75,100,125,150,175,200),
  names = preprocessed_documents$labels)

## End(Not run)
```

---

**Description**

Assessing the effects of preprocessing decisions on an outcome variable.

**Usage**

```r
preprocessing_choice_regression(Y, choices, dataset = "UK",
                               base_case_index = 128)
```

**Arguments**

- `Y` A vector of length 128 (usually) containing a numeric outcome variable. This should be the preText (or other) score for a particular preprocessing specification.
- `choices` A 128 x 7 data.frame produced by the `factorial_preprocessing()` function and output in the ‘$choices’ field.
- `dataset` The name to be given to the data we are analyzing.
- `base_case_index` An optional argument which removes a base case row from the choices data before performing the regression.
Value

A data.frame

Examples

## Not run:
# *** note that this function is already called in the pretext() function and
# its output is returned in the results.
# load the package
library(pretext)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(  
  UK_Manifestos,  
  use_ngrams = TRUE,  
  infrequent_term_threshold = 0.02,  
  verbose = TRUE)
# run pretext
pretext_results <- pretext(  
  preprocessed_documents,  
  dataset_name = "Inaugural Speeches",  
  distance_method = "cosine",  
  num_comparisons = 100,  
  verbose = TRUE)
# get results
reg_results <- preprocessing_choice_regression(  
  pretext_results$pretext_scores$pretext_score,  
  preprocessed_documents$choices,  
  dataset = "UK Manifestos",  
  base_case_index = 128)

## End(Not run)

preText: Diagnostics to Assess The Effects of Text Preprocessing Decisions

Description

preText: Diagnostics to Assess The Effects of Text Preprocessing Decisions

Calculates preText scores for each preprocessing specification.

Usage

preText(preprocessed_documents, dataset_name = "Documents",  
  distance_method = "cosine", num_comparisons = 50, parallel = FALSE,  
  cores = 1, verbose = TRUE)
**Arguments**

- `preprocessed_documents`  
  A list object generated by the `factorial_preprocessing()` function.

- `dataset_name`  
  A string indicating the name to be associated with the results. Defaults to "Documents".

- `distance_method`  
  The method that should be used for calculating document distances. Defaults to "cosine".

- `num_comparisons`  
  If method = "distribution", the number of ranks to use in calculating average difference. Defaults to 50.

- `parallel`  
  Logical indicating whether factorial preprocessing should be performed in parallel. Defaults to FALSE.

- `cores`  
  Defaults to 1, can be set to any number less than or equal to the number of cores on one's computer.

- `verbose`  
  Logical indicating whether more information should be printed to the screen to let the user know about progress. Defaults to TRUE.

**Value**

A result list object.

**preText functions**

To use this package, you will first want to check out the `factorial_preprocessing()` function which will take raw data and transform it into document-frequency matrices using a factorial design and 6-7 different preprocessing decisions. The next step in most applications will be to run the `preText()` function, which will generate preText scores for each preprocessing specification. These can then be fed to the `preText_score_plot()` and `regression_coefficient_plot()` functions to generate interpretable output. For more information on additional functions check out the GitHub README for this package (https://github.com/matthewjdenny/preText) or the "getting started" vignette by typing `vignette("getting_started_with_preText")` into the console.

**Examples**

```r
## Not run:
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
    UK_Manifestos,
    use_ngrams = TRUE,
    infrequent_term_threshold = 0.02,
    verbose = TRUE)
# run preText
preText_results <- preText(
    preprocessed_documents,
```
Description

preText plots for each preprocessing specification.

Usage

\[
\text{preText\_score\_plot}(\text{preText\_results}, \text{display\_raw\_rankings} = \text{FALSE}, \text{remove\_labels} = \text{FALSE}, \text{num\_docs} = \text{NULL}, \text{text\_size} = 1)
\]

Arguments

- \text{preText\_results}
  - The output from the ‘preText\_test()’ or ‘preText()’ functions.

- \text{display\_raw\_rankings}
  - Logical indicating whether raw ranking differences should be displayed (as opposed to relative differences).

- \text{remove\_labels}
  - Option to remove preprocessing step labels. Defaults to FALSE.

- \text{num\_docs}
  - If display\_raw\_rankings = TRUE, the number of documents in the corpus.

- \text{text\_size}
  - The ‘cex’ for text in dot plot generated by function. Defaults to 1.

Value

A plot

Examples

## Not run:
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use\_ngrams = TRUE,
  infrequent\_term\_threshold = 0.02,
  verbose = TRUE)
# run preText
preText_results <- preText(
  preprocessed_documents,
  dataset_name = "Inaugural Speeches",
  distance_method = "cosine",
  num_comparisons = 100,
  verbose = TRUE)
# generate preText score plot
preText_score_plot(preText_results)

## End(Not run)

**Description**
calculates preText scores for each preprocessing specification.

**Usage**
preText_test(distance_matrices, choices, labels = NULL,
             baseline_index = 128, text_size = 1, num_comparisons = 50,
             parallel = FALSE, cores = 1, verbose = TRUE)

**Arguments**
- **distance_matrices**
  A list of document distance matrices generated by the ‘scaling_comparison()’ function and returned in the ‘$distance_matrices’ field.
- **choices**
  A dataframe indicating whether a preprocessing step was applied or not, for each preprocessing step. This is generated by the ‘factorial_preprocessing()’ function and returned in the ‘$choices’ field.
- **labels**
  Optional argument giving names for each preprocessing step. This is generated by the ‘factorial_preprocessing()’ function and returned in the ‘$labels’ field.
- **baseline_index**
  The index of the baseline distance matrix against which we are comparing. Defaults to 128, which is the most minimal preprocessing for our current implementation.
- **text_size**
  The ‘cex’ for text in dot plot generated by function.
- **num_comparisons**
  The number of ranks to use in calculating average difference. Defaults to 50.
- **parallel**
  Logical indicating whether factorial preprocessing should be performed in parallel. Defaults to FALSE.
- **cores**
  Defaults to 1, can be set to any number less than or equal to the number of cores on one’s computer.
- **verbose**
  Logical indicating whether more information should be printed to the screen to let the user know about progress. Defaults to TRUE.
regression_coefficient_plot

Regression Coefficient Plot

Description

Easy plotting for multiple regression results on same axes.

Usage

regression_coefficient_plot(data, text_size = 1, remove_intercept = FALSE,
title = "")
remove_infrequent_terms

Remove infrequently occurring terms from quanteda dfm.

Description

Removes terms appearing in less than a specific proportion of documents in a corpus from a dfm.

Arguments

data: A data.frame produced by the `preprocessing_choice_regression()` function, or a list created by the `preText()` function.
text_size: The size of the text to be displayed. Defaults to 1.
remove_intercept: Logical indicating whether intercept coefficient should be plotted. Defaults to FALSE.
title: The title the user wishes to give the plot, which will be displayed instead of the axis title.

Value

A plot

Examples

## Not run:
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
    UK_Manifestos,
    use_ngrams = TRUE,
    infrequent_term_threshold = 0.02,
    verbose = TRUE)
# run preText
preText_results <- preText(
    preprocessed_documents,
    dataset_name = "Inaugural Speeches",
    distance_method = "cosine",
    num_comparisons = 100,
    verbose = TRUE)
# generate regression results
regression_coefficient_plot(preText_results,
    remove_intercept = TRUE)

## End(Not run)
Usage

```r
remove_infrequent_terms(dfm_object, proportion_threshold = 0.01,
    indices = NULL, verbose = TRUE)
```

Arguments

- `dfm_object`: A quanteda dfm object.
- `proportion_threshold`: Proportion of documents a term must be included in to be included in the dfm.
- `indices`: Defaults to NULL. If not NULL, then it must be a numeric vector specifying the column indices of terms the user would like to remove. Useful for removing specific terms.
- `verbose`: Logical indicating whether more information should be printed to the screen to let the user know about progress in preprocessing. Defaults to TRUE.

Value

A reduced dfm.

Examples

```r
## Not run:
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
    UK_Manifestos,
    use_ngrams = TRUE,
    infrequent_term_threshold = 0.02,
    verbose = TRUE)
updated_dfm <- remove_infrequent_terms(preprocessed_documents$dfm_list[[1]],
    proportion_threshold = 0.5,
    indices = NULL,
    verbose = TRUE)

## End(Not run)
```

Description

Scale each dfm and return a list of distance matrices and scaled document positions.
scaling_comparison

Usage

scaling_comparison(dfm_object_list, dimensions = 2,
                   distance_method = "cosine", verbose = TRUE, cores = 1)

Arguments

dfm_object_list
   A list of quanteda dfm objects returned in the `$dfm_list$ field of the output
   from the `factorial_preprocessing()` function.

dimensions
   The number of dimensions to be used by the multidimensional scaling algo-
   rithm. Defaults to 2.

distance_method
   The method that should be used for calculating document distances. Defaults to
   "cosine".

verbose
   Logical indicating whether more information should be printed to the screen to
   let the user know about progress. Defaults to TRUE.

cores
   The number of cores to be used for parallelization (optional).

Value

A result list object.

Examples

## Not run:
## *** This function is used automatically inside of the preText() function.
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
   UK_Manifestos,
   use_ngrams = TRUE,
   infrequent_term_threshold = 0.02,
   verbose = TRUE)
# scale documents
scaling_results <- scaling_comparison(preprocessed_documents$dfm_list,
                                       dimensions = 2,
                                       distance_method = "cosine",
                                       verbose = TRUE)

## End(Not run)
topic_key_term_plot  
Plot Prevalence of Topic Key Terms

Description

Plotting of key terms across preprocessing decisions.

Usage

```r
topic_key_term_plot(topic_key_term_results, labs, key_term_columns = 2:6, 
custom_col_names = c("Iraq", "Terrorism", "Al Qaeda", "Insurance", 
"Stem Cell"), custom_labels = c("0\%", "<1\%", "1-2\%", "2-3\%", "3-4\%", 
"4-5\%", "5-6\%", "6-7\%", "7-8\%", "8-9\%", "9-10\%", "10\%+"), 
one_matrix = FALSE, thresholds = c(0e-04, 0, 0.0099, 0.0199, 0.0299, 
0.0399, 0.0499, 0.0599, 0.0699, 0.0799, 0.0899, 0.0999), heat_ramp = FALSE, 
return_data = FALSE)
```

Arguments

- `topic_key_term_results`: A data.frame with one column per key term and one row for each set of topic model results. The entries in each cell should be the proportion of topics in which a term appears.
- `labs`: Labels for the preprocessing specifications associated with each set of topic model results.
- `key_term_columns`: The columns containing key term results.
- `custom_col_names`: Names for the key terms.
- `custom_labels`: Labels for the provided key. Must be of length 12.
- `one_matrix`: Logical indicating whether results should be displayed as a one column matrix. Defaults to FALSE.
- `thresholds`: A numeric vector of length 11 with threshold for inclusion in various heat map categories. Defaults to FALSE.
- `heat_ramp`: Option to use heat ramp (yellow-red-purple) instead of a white to blue ramp.
- `return_data`: Logical indicating whether rescaled data should be returned. Defaults to FALSE.

Value

A plot
Examples

```r
# Not run:
set.seed(12345)
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
cross_validation_splits <- 10
# create 10 test/train splits
train_inds <- vector(mode = "list", length = cross_validation_splits)
test_inds <- vector(mode = "list", length = cross_validation_splits)
# sample CV indices
for (i in 1:cross_validation_splits) {
    test <- sample(1:length(UK_Manifestos),
                   size = round(length(UK_Manifestos)/5),
                   replace = FALSE)
    train <- 1:length(UK_Manifestos)
    for (j in 1:length(test)) {
        train[which(train == test[j])]
    }
    train_inds[[i]] <- train
test_inds[[i]] <- test
}
# get the optimal number of topics (this will take a very long time):
optimal_k <- optimal_k_comparison(
  train_inds,
  test_inds,
  preprocessed_documents$dfm_list,
  topics = c(25,50,75,100,125,150,175,200),
  names = preprocessed_documents$labels)
# run a topic model with the optimal number of topics for each preproc. spec.
top_terms_list <- vector(mode = "list", length = 128)
for (i in 1:128) {
    fit <- topicmodels::LDA(quanteda::convert(preprocessed_documents$dfm_list[[i]],
                                              to = "topicmodels"),
                            k = optimal_k[i])
    # extract out top 20 terms for each topic
    top_terms <- terms(fit, 20)
    top_terms_list[[i]] <- top_terms
}
# !!!!! You will need to look for some key terms, and store them in a
# data.frame. Your code should be based off of the following. !!!!
# function to search for a term
find_term <- function(vec, term) {
  tc <- 0
  for (i in 1:length(term)) {
```
tc <- tc + sum(grepl(term[i], vec, ignore.case = T))
}
if (tc > 0) {
  return(TRUE)
} else {
  return(FALSE)
}

# look for topics containing the terms below -- this is from our example with
# press releases so it will have to be modified.
# allows for multiple top terms related to the same concept
num_topics <- rep(0, length = 128)
search_list <- list(iraq = c("iraq"),
  terror = c("terror"),
  al_qaeda = c("qaeda"),
  insurance = c("insur"),
  stem_cell = c("stem"))

# where we will store our results
topics_in_results <- data.frame(
  preprocessing_steps = preprocessed_documents$labels,
  iraq = num_topics,
  terror = num_topics,
  al_qaeda = num_topics,
  insurance = num_topics,
  stem_cell = num_topics,
  optimal_number_of_topics = optimal_k,
  stringsAsFactors = FALSE)

# count the number of topics in which each top term appears
for (i in 1:128) {
  # allows for multiple top terms related to the same concept
top_terms <- top_terms_list[[i]]
  for (j in 1:length(search_list[i])) {
    in_topic <- apply(top_terms, 2, find_term, term = search_list[[j]])
    which_topics <- which(in_topic)
    topics_in_results[i, (j+1)] <- length(which_topics)
  }
}

# now make a plot:
topic_key_term_plot(
  topics_in_results,
  preprocessed_documents$labels,
  key_term_columns = 2:6,
  custom_col_names = c("Iraq", "Terrorism", "Al Qaeda", "Insurance", "Stem Cell"),
  custom_labels = c("%", "<1%", "1-2%", "2-3%", "3-4%", "4-5%", "5-6%", "6-7%", "7-8%", "8-9%", "9-10%", "10+%"),
  one_matrix = FALSE,
  thresholds = c(-0.0001, 0, 0.0099, 0.0199, 0.0299, 0.0399, 0.0499, 0.0599, 0.0699, 0.0799, 0.0899, 0.0999),
  heat_ramp = FALSE,
  return_data = FALSE)
### Description

Calculates the novelty score for a character matrix displaying topic top-terms for all topics.

### Usage

```r
topic_novelty_score(top_terms_matrix, row_range = NULL)
```

### Arguments

- `top_terms_matrix`: A character matrix or data.frame containing top terms for all topics.
- `row_range`: Optional argument specifying range of rows to keep. This is useful if we are only interested in the top 10 or 20 terms, but we have a matrix with the top 50 terms for each topic.

### Value

A novelty score.

### Examples

```r
### Not run:
set.seed(12345)
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
cross_validation_splits <- 10
# create 10 test/train splits
train_inds <- vector(mode = "list", length = cross_validation_splits)
test_inds <- vector(mode = "list", length = cross_validation_splits)
# sample CV indices
for (i in 1:cross_validation_splits) {
  test <- sample(1:length(UK_Manifestos),
                 size = round(length(UK_Manifestos)/5),
                 replace = FALSE)
  train <- 1:length(UK_Manifestos)
  ```
for (j in 1:length(test)) {
  train <- train[which(train == test[j])]
}
train_inds[[i]] <- train
test_inds[[i]] <- test

# get the optimal number of topics (this will take a very long time):
optimal_k <- optimal_k_comparison(
  train_inds,
  test_inds,
  preprocessed_documents$dfm_list,
  topics = c(25,50,75,100,125,150,175,200),
  names = preprocessed_documents$labels)

# run a topic model with the optimal number of topics for each preproc. spec.
top_terms_list <- vector(mode = "list", length = 128)
for (i in 1:128) {
  fit <- topicmodels::LDA(quanteda::convert(preprocessed_documents$dfm_list[[i]],
                                                  to = "topicmodels"),
                          k = optimal_k[i])
  # extract out top 20 terms for each topic
  top_terms <- terms(fit,20)
  top_terms_list[[i]] <- top_terms
}

# calculate novelty score
topic_novelty_score(top_terms_list[[i]])

### End(Not run)

---

**UK_Manifestos**  
*Full text of 69 UK party manifestos from 1918-2001.*

**Description**

A dataset of party manifestos for Liberal, Labour, and Conservative parties for every general election in the UK between 1918 and 2001 (23 elections).

**Usage**

UK_Manifestos

**Format**

A character vector of length 69, with one document per entry.

**Source**

Aggregated from several online resources but see, for example: [http://www.politicsresources.net/area/uk/man.htm](http://www.politicsresources.net/area/uk/man.htm)
wordfish_comparison

Description

Calculated Wordfish scores for a list of dfm objects with temporal filtering.

Usage

wordfish_comparison(dfm_list, years, anchors = c(1, 24), proportion_threshold = 1, document_inindices = NULL)

Arguments

dfm_list A list of quanteda dfm objects generated by the ‘factorial_preprocessing()’ and returned in the ‘$dfm_list’ field
years A numeric vector giving the year for each document.
anchors A numeric vector of length two used to anchor the Wordfish estimates. Defaults to c(1,24) which should work for the UK parliament docs.
proportion_threshold proportion of years a term must be included in to be included in the Wordfish analysis.
document_inindices An option vector of row indices to be used. Useful for using a subset of the data for analysis.

Value

A result list object

Examples

## Not run:
# replicates wordfish analysis from Denny and Spirling (2016)
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
  UK_Manifestos,
  use_ngrams = TRUE,
  infrequent_term_threshold = 0.02,
  verbose = TRUE)
# get the years each document was written and store them as a numeric vector
dfm <- preprocessed_documents$dfm_list[[1]]
rl <- function(str) {
  stringr::str_replace_all(str, "[A-Za-z]+", "")}
years <- as.numeric(sapply(rownames(dfm_list), numeric))

# use the wordfish_comparison function to compare all dfms. We are using
# conservative and labour manifestos from 1983, 1987, 1992, and 1997 for a total
# of 8 manifestos. These are indicated by the document_inidices = c(19:22,42:45)
# argument. You can see the document names by entering rownames(dfm) into the
# console. We need to set the anchors to 5,1 because anchoring is applied in the
# reduced dfm. We are also only including terms that appear at least once in a
# manifesto from each of the 4 years, to deal with the strong temporal effects.
wordfish_results <- wordfish_comparison(
  preprocessed_documents$dfm_list,
  years,
  anchors = c(1,5),
  proportion_threshold = 1,
  document_inidices = c(19:22,42:45))

## End(Not run)

wordfish_rank_plot  Plot of Wordfish rankings of documents

Description

Coloration by ground-truth ranking.

Usage

wordfish_rank_plot(wordfish_results, labels, invert = TRUE,
  "Con1992", "Con1987", "Con1983"), black_white = FALSE, one_matrix = FALSE,
  return_deviations = FALSE)

Arguments

wordfish_results  The output from the ‘wordfish_comparison()’ function.
labels  A character vector giving the names for each preprocessing step.
invert  Logical indicating whether Wordfish score rankings should be reversed internally.
ranking  A character vector containing the correctly ranked document names.
black_white  Logical, defaults to FALSE. If FALSE then results are displayed on a red-blue scale. IF TRUE, then mis-ordered documents are colored black.
one_matrix  Logical indicating whether results should be plotted as a one or two column matrix. Defaults to FALSE.
return_deviations  Return a dataset indicating the ordering deviations for each preprocessing combination. Defaults to FALSE.
**Value**

A plot.

**Examples**

```r
## Not run:
# replicates wordfish analysis from Denny and Spirling (2016)
# load the package
library(preText)
# load in the data
data("UK_Manifestos")
# preprocess data
preprocessed_documents <- factorial_preprocessing(
    UK_Manifestos,
    use_ngrams = TRUE,
    infrequent_term_threshold = 0.02,
    verbose = TRUE)
# get the years each document was written and store them as a numeric vector
dfm <- preprocessed_documents$dfm_list[[1]]
rl <- function(str) {
    stringr::str_replace_all(str,
        "[A-Za-z]+",""
    }
years <- as.numeric(sapply(rownames(dfm),rl))

# use the wordfish_comparison function to compare all dfms. We are using
# conservative and labour manifestos from 1983, 1987, 1992, and 1997 for a total
# of 8 manifestos. These are indicated by the document_inidices = c(19:22,42:45)
# argument. You can see the document names by entering rownames(dfm) into the
# console. We need to set the anchors to 5,1 because anchoring is applied in the
# reduced dfm. We are also only including terms that appear atleast once in a
# manifesto from each of the 4 years, to deal with the strong temporal effects.
wordfish_results <- wordfish_comparison(
    preprocessed_documents$dfm_list,
    years,
    anchors = c(1,5),
    proportion_threshold = 1,
    document_inidices = c(19:22,42:45))
deviations <- wordfish_rank_plot(wordfish_results,
    labels = preprocessed_documents$labels,
    invert = FALSE,
        "Con1997","Con1992","Con1987","Con1983"),
    black_white = FALSE,
    one_matrix = FALSE,
    return_deviations = FALSE)
```

## End(Not run)
Index

*Topic datasets
  UK_Manifestos, 24
  calculate_prediction_errors, 2
  dfm_scaling_test, 3
  document_position_plots, 4
  factorial_preprocessing, 5
  mantel_comparison, 7
  mantel_comparison_to_base, 8
  optimal_k_comparison, 9
  preprocessing_choice_regression, 11
  preText, 12
  preText-package (preText), 12
  preText_score_plot, 14
  preText_test, 15
  regression_coefficient_plot, 16
  remove_infrequent_terms, 17
  scaling_comparison, 18
  topic_key_term_plot, 20
  topic_novelty_score, 23
  UK_Manifestos, 24
  wordfish_comparison, 25
  wordfish_rank_plot, 26