Package ‘quanteda.textmodels’

October 13, 2022

Type Package

Title Scaling Models and Classifiers for Textual Data

Version 0.9.5-1


Depends R (>= 3.1.0), methods

Imports glmnet, LiblineaR, Matrix (>= 1.2), quanteda (>= 2.0), RSpectra, Rcpp (> 0.12.12), RcppParallel, SparseM, stringi

Suggests ca, covr, fastNaiveBayes, knitr, lsa, microbenchmark, naivebayes, quanteda.textplots, spelling, testthat, rmarkdown

LinkingTo Rcpp, RcppParallel, RcppArmadillo (>= 0.7.600.1.0), quanteda

URL https://github.com/quanteda/quanteda.textmodels

License GPL-3

Encoding UTF-8

LazyData true

Language en-GB

RoxygenNote 7.2.1

SystemRequirements C++11


VignetteBuilder knitr
data_corpus_dailnoconf1991

Confidence debate from 1991 Irish Parliament

Description

Texts of speeches from a no-confidence motion debated in the Irish Dáil from 16-18 October 1991 over the future of the Fianna Fail-Progressive Democrat coalition. (See Laver and Benoit 2002 for details.)

Usage

data_corpus_dailnoconf1991
data_corpus_EPcoaldebate

Format
data_corpus_dailnoconf1991 is a corpus with 58 texts, including docvars for name, party, and position.

Source

References


Examples
```r
## Not run:
library("quanteda")
data_dfm_dailnoconf1991 <- data_corpus_dailnoconf1991 %>%
  tokens(remove_punct = TRUE) %>%
dfm() tmod <- textmodel_affinity(data_dfm_dailnoconf1991, c("Govt", "Opp", "Opp", rep(NA, 55)))
(pred <- predict(tmod))
dat <-
data.frame(party = as.character(docvars(data_corpus_dailnoconf1991, "party")),
  govt = coef(pred)[, "Govt"],
  position = as.character(docvars(data_corpus_dailnoconf1991, "position")),
  stringsAsFactors = FALSE)
  bymedian <- with(dat, reorder(paste(party, position), govt, median))
par(mar = c(5, 6, 4, 2)+.1)
boxplot(govt ~ bymedian, data = dat,
  horizontal = TRUE, las = 1,
  xlab = "Degree of support for government")
abline(h = 7.5, col = "red", lty = "dashed")
text(c(0.9, 0.9), c(8.5, 6.5), c("Government", "Opposition"))
## End(Not run)
```

data_corpus_EPcoaldebate
Crowd-labelled sentence corpus from a 2010 EP debate on coal subsidies
Description

A multilingual text corpus of speeches from a European Parliament debate on coal subsidies in 2010, with individual crowd codings as the unit of observation. The sentences are drawn from officially translated speeches from a debate over a European Parliament debate concerning a Commission report proposing an extension to a regulation permitting state aid to uncompetitive coal mines.

Each speech is available in six languages: English, German, Greek, Italian, Polish and Spanish. The unit of observation is the individual crowd coding of each natural sentence. For more information on the coding approach see Benoit et al. (2016).

Usage

data_corpus_EPcoaldebate

Format

The corpus consists of 16,806 documents (i.e. codings of a sentence) and includes the following document-level variables:

sentence_id character; a unique identifier for each sentence
crowd_subsidy_label factor; whether a coder labelled the sentence as "Pro-Subsidy", "Anti-Subsidy" or "Neutral or inapplicable"
language factor; the language (translation) of the speech
name_last character; speaker’s last name
name_first character; speaker’s first name
ep_group factor; abbreviation of the EP party group of the speaker
country factor; the speaker’s country of origin
vote factor; the speaker’s vote on the proposal (For/Against/Abstain/NA)
coder_id character; a unique identifier for each crowd coder
coder_trust numeric; the "trust score" from the Crowdflower platform used to code the sentences, which can theoretically range between 0 and 1. Only coders with trust scores above 0.8 are included in the corpus.

A corpus object.

References

**data_corpus_irishbudget2010**

*Irish budget speeches from 2010*

**Description**

Speeches and document-level variables from the debate over the Irish budget of 2010.

**Usage**

```
data_corpus_irishbudget2010
```

**Format**

The corpus object for the 2010 budget speeches, with document-level variables for year, debate, serial number, first and last name of the speaker, and the speaker’s party.

**Details**

At the time of the debate, Fianna Fáil (FF) and the Greens formed the government coalition, while Fine Gael (FG), Labour (LAB), and Sinn Féin (SF) were in opposition.

**Source**


**References**


---

**data_corpus_moviereviews**

*Movie reviews with polarity from Pang and Lee (2004)*

**Description**

A corpus object containing 2,000 movie reviews classified by positive or negative sentiment.

**Usage**

```
data_corpus_moviereviews
```
Format

The corpus includes the following document variables:

- **sentiment**: factor indicating whether a review was manually classified as positive `pos` or negative `neg`.
- **id1**: Character counting the position in the corpus.
- **id2**: Random number for each review.

Details

For more information, see `cat(meta(data_corpus_moviesreviews, "readme"))`.

Source

[https://www.cs.cornell.edu/people/pabo/movie-review-data/](https://www.cs.cornell.edu/people/pabo/movie-review-data/)

References


Examples

```r
# check polarities
table(data_corpus_moviesreviews$sentiment)

# make the data into sentences, because each line is a sentence
data_moviesreviewsents <-
  quanteda::corpus_segment(data_moviesreviews, "\n", extract_pattern = FALSE)
print(data_moviesreviewsents, max_ndoc = 3)
```

```
textmodel_affinity  Class affinity maximum likelihood text scaling model

Description

textmodel_affinity implements the maximum likelihood supervised text scaling method described in Perry and Benoit (2017).

Usage

textmodel_affinity(
  x,
  y,
  exclude = NULL,
  smooth = 0.5,
  ref_smooth = 0.5,
  verbose = quanteda_options("verbose")
)
```
textmodel_ca

Correspondence analysis of a document-feature matrix

Description

textmodel_ca implements correspondence analysis scaling on a dfm. The method is a fast/sparse version of function ca.

Arguments

x   the dfm or bootstrap_dfm object on which the model will be fit. Does not need to contain only the training documents, since the index of these will be matched automatically.
y   vector of training classes/scores associated with each document identified in data
exclude   a set of words to exclude from the model
smooth   a smoothing parameter for class affinities; defaults to 0.5 (Jeffreys prior). A plausible alternative would be 1.0 (Laplace prior).
ref_smooth   a smoothing parameter for token distributions; defaults to 0.5
verbose   logical; if TRUE print diagnostic information during fitting.

Author(s)

Patrick Perry and Kenneth Benoit

References


See Also

predict.textmodel_affinity() for methods of applying a fitted textmodel_affinity model object to predict quantities from (other) documents.

Examples

(af <- textmodel_affinity(quanteda::data_dfm_lbgexample, y = c("L", NA, NA, NA, "R", NA)))
predict(af)
predict(af, newdata = quanteda::data_dfm_lbgexample[6, ])

## Not run:
# compute bootstrapped SEs
dfmat <- quanteda::bootstrap_dfm(data_corpus_dailnoconf1991, n = 10, remove_punct = TRUE)
textmodel_affinity(dfmat, y = c("Govt", "Opp", "Opp", rep(NA, 55)))

## End(Not run)


textmodel_ca

Usage

textmodel_ca(x, smooth = 0, nd = NA, sparse = FALSE, residual_floor = 0.1)

Arguments

x
  the dfm on which the model will be fit
smooth
  a smoothing parameter for word counts; defaults to zero.
nd
  Number of dimensions to be included in output; if NA (the default) then the
  maximum possible dimensions are included.
sparse
  retains the sparsity if set to TRUE; set it to TRUE if x (the dfm) is too big to be
  allocated after converting to dense
residual_floor
  specifies the threshold for the residual matrix for calculating the truncated svd.
  Larger value will reduce memory and time cost but might reduce accuracy; only appli-
  cable when sparse = TRUE

Details

svds in the RSpectra package is applied to enable the fast computation of the SVD.

Value

textmodel_ca() returns a fitted CA textmodel that is a special class of ca object.

Note

You may need to set sparse = TRUE and increase the value of residual_floor to ignore less
important information and hence to reduce the memory cost when you have a very big dfm. If
your attempt to fit the model fails due to the matrix being too large, this is probably because of the
memory demands of computing the $V \times V$ residual matrix. To avoid this, consider increasing the
value of residual_floor by 0.1, until the model can be fit.

Author(s)

Kenneth Benoit and Haiyan Wang

References

Nenadic, O. & Greenacre, M. (2007). Correspondence Analysis in R, with Two- and Three-
jss.v020.i03

See Also

coeff.textmodel_lsa(), ca
Examples

library("quanteda")
dfmat <- dfm(tokens(data_corpus_irishbudget2010))
tmod <- textmodel_ca(dfmat)
summary(tmod)

textmodel_lr

Logistic regression classifier for texts

Description

Fits a fast penalized maximum likelihood estimator to predict discrete categories from sparse dfm objects. Using the \texttt{glmnet} package, the function computes the regularization path for the lasso or elasticnet penalty at a grid of values for the regularization parameter lambda. This is done automatically by testing on several folds of the data at estimation time.

Usage

\texttt{textmodel\_lr(x, y, \ldots)}

Arguments

\begin{itemize}
\item \texttt{x} \hspace{1cm} the \texttt{dfm} on which the model will be fit. Does not need to contain only the training documents.
\item \texttt{y} \hspace{1cm} vector of training labels associated with each document identified in \texttt{train}. (These will be converted to factors if not already factors.)
\item \ldots \hspace{1cm} additional arguments passed to \texttt{cv.glmnet()}
\end{itemize}

References


See Also

\texttt{cv.glmnet()}, \texttt{predict.textmodel\_lr()}, \texttt{coef.textmodel\_lr()}

Examples

\begin{verbatim}
## Example from 13.1 of _An Introduction to Information Retrieval_
library("quanteda")
corp <- corpus(c(d1 = "Chinese Beijing Chinese",
               d2 = "Chinese Chinese Shanghai",
               d3 = "Chinese Macao",
               d4 = "Tokyo Japan Chinese",
               d5 = "London England Chinese",
               d6 = "Chinese Chinese Chinese Tokyo Japan"),
               docvars = data.frame(train = factor(c("Y", "Y", "Y", "N", "N", NA))))
\end{verbatim}
dfmat <- dfm(tokens(corp), tolower = FALSE)

## simulate bigger sample as classification on small samples is problematic
set.seed(1)
dfmat <- dfm_sample(dfmat, 50, replace = TRUE)

## train model
(tmodl <- textmodel_lr(dfmat, docvars(dfmat, "train")))
summary(tmodl)
coef(tmodl)

## predict probability and classes
predict(tmodl, type = "prob")
predict(tmodl)

---

**textmodel_lsa**

**Latent Semantic Analysis**

**Description**

Fit the Latent Semantic Analysis scaling model to a dfm, which may be weighted (for instance using `quanteda::dfm_tfidf()`).

**Usage**

`textmodel_lsa(x, nd = 10, margin = c("both", "documents", "features"))`

**Arguments**

- `x` the dfm on which the model will be fit
- `nd` the number of dimensions to be included in output
- `margin` margin to be smoothed by the SVD

**Details**

`svds` in the `RSpectra` package is applied to enable the fast computation of the SVD.

**Note**

The number of dimensions `nd` retained in LSA is an empirical issue. While a reduction in `k` can remove much of the noise, keeping too few dimensions or factors may lose important information.

**Author(s)**

Haiyan Wang and Kohei Watanabe
textmodel_nb

Naive Bayes classifier for texts

Description

Fit a multinomial or Bernoulli Naive Bayes model, given a dfm and some training labels.

Usage

textmodel_nb(
  x,
  y,
  smooth = 1,
  prior = c("uniform", "docfreq", "termfreq"),
  distribution = c("multinomial", "Bernoulli")
)

References


See Also

predict.textmodel_lsa(), coef.textmodel_lsa()
Arguments

- **x**: the dfm on which the model will be fit. Does not need to contain only the training documents.
- **y**: vector of training labels associated with each document identified in `train`. (These will be converted to factors if not already factors.)
- **smooth**: smoothing parameter for feature counts, added to the feature frequency totals by training class
- **prior**: prior distribution on texts; one of "uniform", "docfreq", or "termfreq". See Prior Distributions below.
- **distribution**: count model for text features, can be multinomial or Bernoulli. To fit a "binary multinomial" model, first convert the dfm to a binary matrix using `quanteda::dfm_weight(x, scheme = "boolean")`.

Value

textmodel_nb() returns a list consisting of the following (where *I* is the total number of documents, *J* is the total number of features, and *k* is the total number of training classes):

- **call**: original function call
- **param**: *k* × *V*; class conditional posterior estimates
- **x**: the *N* × *V* training dfm *x*
- **y**: the *N*-length y training class vector, where NAs will not be used will be retained in the saved x matrix
- **distribution**: character; the distribution of x for the NB model
- **priors**: numeric; the class prior probabilities
- **smooth**: numeric; the value of the smoothing parameter

Prior distributions

Prior distributions refer to the prior probabilities assigned to the training classes, and the choice of prior distribution affects the calculation of the fitted probabilities. The default is uniform priors, which sets the unconditional probability of observing the one class to be the same as observing any other class.

"Document frequency" means that the class priors will be taken from the relative proportions of the class documents used in the training set. This approach is so common that it is assumed in many examples, such as the worked example from Manning, Raghavan, and Schütze (2008) below. It is not the default in `quanteda`, however, since there may be nothing informative in the relative numbers of documents used to train a classifier other than the relative availability of the documents. When training classes are balanced in their number of documents (usually advisable), however, then the empirically computed "docfreq" would be equivalent to "uniform" priors.

Setting **prior** to "termfreq" makes the priors equal to the proportions of total feature counts found in the grouped documents in each training class, so that the classes with the largest number of features are assigned the largest priors. If the total count of features in each training class was the same, then "uniform" and "termfreq" would be the same.
Smoothing parameter

The smooth value is added to the feature frequencies, aggregated by training class, to avoid zero frequencies in any class. This has the effect of giving more weight to infrequent term occurrences.

Author(s)

Kenneth Benoit

References


See Also

`predict.textmodel_nb()`

Examples

```r
## Example from 13.1 of _An Introduction to Information Retrieval_
library("quanteda")
txt <- c(d1 = "Chinese Beijing Chinese",
       d2 = "Chinese Chinese Shanghai",
       d3 = "Chinese Macao",
       d4 = "Tokyo Japan Chinese",
       d5 = "Chinese Chinese Chinese Tokyo Japan")
x <- dfm(tokens(txt), tolower = FALSE)
y <- factor(c("Y", "Y", "Y", "N", NA), ordered = TRUE)
## replicate IIR p261 prediction for test set (document 5)
(tmod1 <- textmodel_nb(x, y, prior = "docfreq"))
summary(tmod1)
coef(tmod1)
predict(tmod1, type = "prob")
predict(tmod1)

# contrast with other priors
predict(textmodel_nb(x, y, prior = "uniform"))
predict(textmodel_nb(x, y, prior = "termfreq"))

## replicate IIR p264 Bernoulli Naive Bayes
tmod2 <- textmodel_nb(x, y, distribution = "Bernoulli", prior = "docfreq")
predict(tmod2, newdata = x[5, ], type = "prob")
predict(tmod2, newdata = x[5, ])
```
textmodel_svm

Linear SVM classifier for texts

Description

Fit a fast linear SVM classifier for texts, using the LiblineaR package.

Usage

textmodel_svm(
  x,
  y,
  weight = c("uniform", "docfreq", "termfreq"),
  type = 1,
  ...
)

Arguments

x the dfm on which the model will be fit. Does not need to contain only the training documents.

y vector of training labels associated with each document identified in train. (These will be converted to factors if not already factors.)

weight weights for different classes for imbalanced training sets, passed to wi in LiblineaR::LiblineaR(). "uniform" uses default; "docfreq" weights by the number of training examples, and "termfreq" by the relative sizes of the training classes in terms of their total lengths in tokens.

type argument passed to the type argument in LiblineaR::LiblineaR(); default is 1 for L2-regularized L2-loss support vector classification (dual)

... additional arguments passed to LiblineaR::LiblineaR()

References


See Also

LiblineaR::LiblineaR() predict.textmodel_svm()

Examples

# use party leaders for govt and opposition classes
library("quanteda")
docvars(data_corpus_irishbudget2010, "govtopp") <-
dfmat <- dfm(tokens(data_corpus_irishbudget2010))
tmod <- textmodel_svm(dfmat, y = dfmat$govtopp)
predict(tmod)

# multiclass problem - all party leaders
tmod2 <- textmodel_svm(dfmat, 
y = c(rep(NA, 3), "SF", "FF", "FG", NA, "LAB", NA, NA, "Green", rep(NA, 3))
predict(tmod2)

textmodel_wordfish  Wordfish text model

Description


Usage

textmodel_wordfish(
  x,
  dir = c(1, 2),
  priors = c(Inf, Inf, 3, 1),
  tol = c(1e-06, 1e-08),
  dispersion = c("poisson", "quasipoisson"),
  dispersion_level = c("feature", "overall"),
  dispersion_floor = 0,
  sparse = FALSE,
  abs_err = FALSE,
  svd_sparse = TRUE,
  residual_floor = 0.5
)

Arguments

x  the dfm on which the model will be fit

dir  set global identification by specifying the indexes for a pair of documents such that $\hat{\theta}_{dir[1]} < \hat{\theta}_{dir[2]}$.

priors  prior precisions for the estimated parameters $\alpha_i, \psi_j, \beta_j$, and $\theta_i$, where $i$ indexes documents and $j$ indexes features

tol  tolerances for convergence. The first value is a convergence threshold for the log-posterior of the model, the second value is the tolerance in the difference in parameter values from the iterative conditional maximum likelihood (from conditionally estimating document-level, then feature-level parameters).

dispersion  sets whether a quasi-Poisson quasi-likelihood should be used based on a single dispersion parameter ("poisson"), or quasi-Poisson ("quasipoisson")
displacement_level
sets the unit level for the displacement parameter, options are "feature" for term-level variances, or "overall" for a single dispersion parameter

dispersion_floor
constraint for the minimal underdispersion multiplier in the quasi-Poisson model. Used to minimize the distorting effect of terms with rare term or document frequencies that appear to be severely underdispersed. Default is 0, but this only applies if dispersion = "quasipoisson".

sparse
specifies whether the "dfm" is coerced to dense. While setting this to TRUE will make it possible to handle larger dfm objects (and make execution faster), it will generate slightly different results each time, because the sparse SVD routine has a stochastic element.

abs_err
specifies how the convergence is considered

svd_spare
uses svd to initialize the starting values of theta, only applies when sparse = TRUE

residual_floor
specifies the threshold for residual matrix when calculating the svds, only applies when sparse = TRUE

Details
The returns match those of Will Lowe's R implementation of wordfish (see the austin package), except that here we have renamed words to be features. (This return list may change.) We have also followed the practice begun with Slapin and Proksch's early implementation of the model that used a regularization parameter of se(σ) = 3, through the third element in priors.

Value
An object of class textmodel_fitted_wordfish. This is a list containing:

- dir global identification of the dimension
- theta estimated document positions
- alpha estimated document fixed effects
- beta estimated feature marginal effects
- psi estimated word fixed effects
- docs document labels
- features feature labels
- sigma regularization parameter for betas in Poisson form
- ll log likelihood at convergence
- se.theta standard errors for theta-hats
- x dfm to which the model was fit

Note
In the rare situation where a warning message of "The algorithm did not converge." shows up, removing some documents may work.
Author(s)

Benjamin Lauderdale, Haiyan Wang, and Kenneth Benoit

References


See Also

`predict.textmodel_wordfish()`

Examples

```r
(tmod1 <- textmodel_wordfish(quanteda::data_dfm_lbgexample, dir = c(1,5)))
summary(tmod1, n = 10)
coef(tmod1)
predict(tmod1)
predict(tmod1, se.fit = TRUE)
predict(tmod1, interval = "confidence")

## Not run:
library("quanteda")
dfmat <- dfm(tokens(data_corpus_irishbudget2010))
(tmod2 <- textmodel_wordfish(dfmat, dir = c(6,5)))
(tmod3 <- textmodel_wordfish(dfmat, dir = c(6,5),
  dispersion = "quasipoisson", dispersion_floor = 0))
(tmod4 <- textmodel_wordfish(dfmat, dir = c(6,5),
  dispersion = "quasipoisson", dispersion_floor = .5))
plot(tmod3$phi, tmod4$phi, xlab = "Min underdispersion = 0", ylab = "Min underdispersion = .5",
  xlim = c(0,1.0), ylim = c(0,1.0))
plot(tmod3$phi, tmod4$phi, xlab = "Min underdispersion = 0", ylab = "Min underdispersion = .5",
  xlim = c(0,1.0), ylim = c(0,1.0), type = "n")
underdispersedTerms <- sample(which(tmod3$phi < 1.0), 5)
which(featnames(dfmat) %in% names(topfeatures(dfmat, 20)))
text(tmod3$phi, tmod4$phi, tmod3$features,
  cex = .8, xlim = c(0,1.0), ylim = c(0,1.0), col = "grey90")
text(tmod3$phi["underdispersedTerms"], tmod4$phi["underdispersedTerms"],
  tmod3$features["underdispersedTerms"],
  cex = .8, xlim = c(0,1.0), ylim = c(0,1.0), col = "black")
if (requireNamespace("austin")) {
  tmod5 <- austin::wordfish(quanteda::as.wfm(dfmat), dir = c(6,5))
cor(tmod1$theta, tmod5$theta)
}
## End(Not run)
```
Description

textmodel_wordscores implements Laver, Benoit and Garry's (2003) "Wordscores" method for scaling texts on a single dimension, given a set of anchoring or reference texts whose values are set through reference scores. This scale can be fitted in the linear space (as per LBG 2003) or in the logit space (as per Beauchamp 2012). Estimates of virgin or unknown texts are obtained using the predict() method to score documents from a fitted textmodel_wordscores object.

Usage

textmodel_wordscores(x, y, scale = c("linear", "logit"), smooth = 0)

Arguments

x the dfm on which the model will be trained
y vector of training scores associated with each document in x
scale scale on which to score the words; "linear" for classic LBG linear posterior weighted word class differences, or "logit" for log posterior differences
smooth a smoothing parameter for word counts; defaults to zero to match the LBG (2003) method. See Value below for additional information on the behaviour of this argument.

Details

The textmodel_wordscores() function and the associated predict() method are designed to function in the same manner as stats::predict.lm(). coef() can also be used to extract the word coefficients from the fitted textmodel_wordscores object, and summary() will print a nice summary of the fitted object.

Value

A fitted textmodel_wordscores object. This object will contain a copy of the input data, but in its original form without any smoothing applied. Calling predict.textmodel_wordscores() on this object without specifying a value for newdata, for instance, will predict on the unsmoothed object. This behaviour differs from versions of quanteda <= 1.2.

Author(s)

Kenneth Benoit
References


See Also

predict.textmodel_wordscores() for methods of applying a fitted textmodel_wordscores model object to predict quantities from (other) documents.

Examples

```r
(tmod <- textmodel_wordscores(quanteda::data_dfm_lbgexample, y = c(seq(-1.5, 1.5, .75), NA)))
summary(tmod)
coef(tmod)
predict(tmod)
predict(tmod, rescaling = "lbg")
predict(tmod, se.fit = TRUE, interval = "confidence", rescaling = "mv")
```
Index

* data
  data_corpus_dailnoconf1991, 2
  data_corpus_EPcoaldebate, 3
  data_corpus_irishbudget2010, 5
  data_corpus_moviereviews, 5
* experimental
  textmodel_affinity, 6
  textmodel_lsa, 10
* textmodel
  textmodel_affinity, 6
  textmodel_lsa, 10
bootstrap_dfm, 7
ca, 7, 8
coef.textmodel_lr(), 9
corr.textmodel_lsa(), 8, 11
corpus, 4
cv.glmnet(), 9
data_corpus_dailnoconf1991, 2
data_corpus_EPcoaldebate, 3
data_corpus_irishbudget2010, 5
data_corpus_moviereviews, 5
dfm, 7–10, 12, 14, 18
LiblineaR::LiblineaR(), 14
predict(), 18
predict.textmodel_affinity(), 7
predict.textmodel_lr(), 9
predict.textmodel_lsa(), 11
predict.textmodel_nb(), 13
predict.textmodel_svm(), 14
predict.textmodel_wordfish(), 17
predict.textmodel_wordscores(), 18, 19
quanteda::dfm_tfidf(), 10
stats::predict.lm(), 18
svds, 8, 10