Package ‘tidylda’

July 19, 2021

Type Package

Title Latent Dirichlet Allocation Using 'tidyverse' Conventions

Version 0.0.1


License MIT + file LICENSE

URL https://github.com/TommyJones/tidylda/

BugReports https://github.com/TommyJones/tidylda/issues

Depends

Imports dplyr, generics, gtools, Matrix, methods, mvrsquared (>= 0.1.0), Rcpp (>= 1.0.2), rlang, stats, stringr, textmineR, tibble, tidy, tidytext

Suggests knitr, parallel, quanteda, testthat, tm, spelling

LinkingTo Rcpp, RcppArmadillo, RcppProgress, RcppThread

Encoding UTF-8

RoxygenNote 7.1.1

Language en-US

SystemRequirements C++11

NeedsCompilation yes

Author Tommy Jones [aut, cre] (<https://orcid.org/0000-0001-6457-2452>), Brendan Knapp [ctb] (<https://orcid.org/0000-0003-3284-4972>), Barum Park [ctb]
Description

augment appends observation level model outputs.

Usage

## S3 method for class 'tidylda'
augment(x, data, type = c("class", "prob"), ...)

Arguments

x | an object of class tidylda
data | a tidy tibble containing one row per original document-token pair, such as is returned by tdm_tidiers with column names c("document", "term") at a minimum.
type | one of either "class" or "prob"
... | other arguments passed to methods, currently not used

Details

The key statistic for augment is \( P(\text{topic} \mid \text{document}, \text{token}) = P(\text{topic} \mid \text{token}) \times P(\text{token} \mid \text{document}) \). \( P(\text{topic} \mid \text{token}) \) are the entries of the 'lambda' matrix in the tidylda object passed with x. \( P(\text{token} \mid \text{document}) \) is taken to be the frequency of each token normalized within each document.
Value

augment returns a tidy tibble containing one row per document-token pair, with one or more columns appended, depending on the value of type.

If type = 'prob', then one column per topic is appended. Its value is P(topic | document, token).

If type = 'class', then the most-probable topic for each document-token pair is returned. If multiple topics are equally probable, then the topic with the smallest index is returned by default.

Description

Calculates the probabilistic coherence of a topic or topics. This approximates semantic coherence or human understandability of a topic.

Usage

calc_prob_coherence(beta, data, m = 5)

Arguments

- **beta**: A numeric matrix or a numeric vector. The vector, or rows of the matrix represent the numeric relationship between topic(s) and terms. For example, this relationship may be p(word|topic) or p(topic|word).

- **data**: A document term matrix or term co-occurrence matrix. The preferred class is a dgCMatrix-class. However there is support for any Matrix-class object as well as several other commonly-used classes such as matrix, dfm, DocumentTermMatrix, and simple_triplet_matrix.

- **m**: An integer for the number of words to be used in the calculation. Defaults to 5

Details

For each pair of words a, b in the top M words in a topic, probabilistic coherence calculates P(b|a) - P(b), where a is more probable than b in the topic. For example, suppose the top 4 words in a topic are a, b, c, d. Then, we calculate 1. P(alb) - P(b), P(alc) - P(c), P(ald) - P(d) 2. P(blac) - P(c), P(bld) - P(d) 3. P(clad) - P(d) All 6 differences are averaged together.

Value

Returns an object of class numeric corresponding to the probabilistic coherence of the input topic(s).
Examples

```r
# Load a pre-formatted dtm and topic model
data(nih_sample_dtm, package = "textmineR")

# fit a model
set.seed(12345)
model <- tidylda(
data = nih_sample_dtm[1:20, ], k = 5,
iterations = 100, burnin = 50
)

calc_prob_coherence(beta = model$beta, data = nih_sample_dtm, m = 5)
```

---

**glance.tidylda**

*Glance method for tidylda objects*

**Description**

`glance` constructs a single-row summary "glance" of a tidylda topic model.

**Usage**

```r
## S3 method for class 'tidylda'

```

**Arguments**

- `x`: an object of class `tidylda`
- `...`: other arguments passed to methods, currently not used

**Value**

`glance` returns a one-row tibble with the following columns:

- `num_topics`: the number of topics in the model
- `num_documents`: the number of documents used for fitting
- `num_tokens`: the number of tokens covered by the model
- `iterations`: number of total Gibbs iterations run
- `burnin`: number of burn-in Gibbs iterations run

**Examples**

```r
dtm <- textmineR::nih_sample_dtm

lda <- tidylda(data = dtm, k = 10, iterations = 100, burnin = 75)

glance(lda)
```
posterior

Draw from the marginal posteriors of a tidylda topic model

Description

These functions are used to sample from the marginal posteriors of a tidylda topic model. This is useful for quantifying uncertainty around the parameters of beta or theta.

Usage

posterior(x, ...)

## S3 method for class 'tidylda'
posterior(x, ...)

## S3 method for class 'tidylda_posterior'
generate(x, matrix, which, times, ...)

Arguments

x
For posterior, an object of class tidylda. For generate, an object of class tidylda_posterior obtained by a call to posterior.

... Other arguments, currently not used.

matrix A character of either 'theta' or 'beta', indicating from which matrix to draw posterior samples.

which Row index of theta, for document, or beta, for topic, from which to draw samples. which may also be a vector of indices.

times Integer number of samples to draw.

Details

To sample from the marginal posteriors of a model, you must first make a call to posterior and then a call to generate.

posterior takes an object of class tidylda and constructs an object of class tidylda_posterior which contains two matrices. The rows of these matrices are Dirichlet parameters used to sample from the marginal posteriors of theta and beta.

generate takes an object of class tidylda_posterior and samples from the marginal posterior of the parameters specified by the matrix argument.

Value

posterior returns an object of class tidylda_posterior.

generate returns a tibble with one row per parameter per sample.

Returns a data frame where each row is a single sample from the posterior. Each column is the distribution over a single parameter. The variable var is a facet for subsetting by document (for theta) or topic (for beta).
predict.tidylda

Get predictions from a Latent Dirichlet Allocation model

Description

Obtains predictions of topics for new documents from a fitted LDA model

Usage

```r
## S3 method for class 'tidylda'
predict( 
```
predict.tidylda

object,
new_data,
method = c("gibbs", "dot"),
iterations = NULL,
burnin = -1,
no_common_tokens = c("default", "zero", "uniform"),
threads = 1,
verbose = TRUE,
...
)

Arguments

object
  a fitted object of class tidylda
new_data
  a DTM or TCM of class dgCMatrix or a numeric vector
method
  one of either "gibbs" or "dot". If "gibbs" Gibbs sampling is used and iterations
  must be specified.
iterations
  If method = "gibbs", an integer number of iterations for the Gibbs sampler to
  run. A future version may include automatic stopping criteria.
burnin
  If method = "gibbs" an integer number of burnin iterations. If burnin is greater
  than -1, the entries of the resulting "theta" matrix are an average over all itera-
  tions greater than burnin. Behavior is the same as documented in tidylda.
no_common_tokens
  behavior when encountering documents that have no tokens in common with the
  model. Options are "default", "zero", or "uniform". See 'details', below for
  explanation of behavior.
threads
  Number of parallel threads, defaults to 1.
verbose
  Logical. Do you want to print a progress bar out to the console? Only active if
  method = "gibbs". Defaults to TRUE.
...
  Additional arguments, currently unused

Details

If predict.tidylda encounters documents that have no tokens in common with the model in
object it will engage in one of three behaviors based on the setting of no_common_tokens.

default (the default) sets all topics to 0 for offending documents. This enables continued com-
putations downstream in a way that NA would not. However, if no_common_tokens == "default",
then predict.tidylda will emit a warning for every such document it encounters.

zero has the same behavior as default but it emits a message instead of a warning.

uniform sets all topics to 1/k for every topic for offending documents. It does not emit a warning
or message.

Value

a "theta" matrix with one row per document and one column per topic
Examples

```r
# load some data
data(nih_sample_dtm, package = "textmineR")

# fit a model
set.seed(12345)

m <- tidylda(
  data = nih_sample_dtm[1:20, ], k = 5,
  iterations = 200, burnin = 175
)

str(m)

# predict on held-out documents using gibbs sampling "fold in"
p1 <- predict(m, nih_sample_dtm[21:100, ],
  method = "gibbs",
  iterations = 200, burnin = 175
)

# predict on held-out documents using the dot product method
p2 <- predict(m, nih_sample_dtm[21:100, ], method = "dot")

# compare the methods
barplot(rbind(p1[1, ], p2[1, ]), beside = TRUE, col = c("red", "blue"))
```

print.tidylda

Print Method for tidylda

Description

Print a summary for objects of class tidylda

Usage

```r
## S3 method for class 'tidylda'
print(x, digits = max(3L, getOption("digits") - 3L), n = 5, ...)
```

Arguments

- `x`: an object of class tidylda
- `digits`: minimal number of significant digits
- `n`: Number of rows to show in each displayed tibble.
- `...`: further arguments passed to or from other methods
Value

Silently returns x

Examples

dtm <- textmineR::nih_sample_dtm

lda <- tidylda(data = dtm, k = 10, iterations = 100)

print(lda)

lda

print(lda, digits = 2)

refit.tidylda

Update a Latent Dirichlet Allocation topic model

Description

Update an LDA model using collapsed Gibbs sampling.

Usage

## S3 method for class 'tidylda'
refit(
  object,
  new_data,
  iterations = NULL,
  burnin = -1,
  prior_weight = 1,
  additional_k = 0,
  optimize_alpha = FALSE,
  calc_likelhood = FALSE,
  calc_r2 = FALSE,
  return_data = FALSE,
  threads = 1,
  verbose = TRUE,
  ...
)

Arguments

object a fitted object of class tidylda.

new_data A document term matrix or term co-occurrence matrix of class dgCMatrix.
refit.tidylda

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterations</td>
<td>Integer number of iterations for the Gibbs sampler to run.</td>
</tr>
<tr>
<td>burnin</td>
<td>Integer number of burnin iterations. If burnin is greater than -1, the resulting &quot;beta&quot; and &quot;theta&quot; matrices are an average over all iterations greater than burnin.</td>
</tr>
<tr>
<td>prior_weight</td>
<td>Numeric, 0 or greater or NA. The weight of the beta as a prior from the base model. See Details, below.</td>
</tr>
<tr>
<td>additional_k</td>
<td>Integer number of topics to add, defaults to 0.</td>
</tr>
<tr>
<td>optimize_alpha</td>
<td>Logical. Do you want to optimize alpha every iteration? Defaults to FALSE. See 'details' of documentation for FitLdaModel for more information.</td>
</tr>
<tr>
<td>calc_likelihood</td>
<td>Logical. Do you want to calculate the log likelihood every iteration? Useful for assessing convergence. Defaults to FALSE.</td>
</tr>
<tr>
<td>calc_r2</td>
<td>Logical. Do you want to calculate R-squared after the model is trained? Defaults to FALSE. This calls CalcTopicModelR2.</td>
</tr>
<tr>
<td>return_data</td>
<td>Logical. Do you want new_data returned as part of the model object?</td>
</tr>
<tr>
<td>threads</td>
<td>Number of parallel threads, defaults to 1.</td>
</tr>
<tr>
<td>verbose</td>
<td>Logical. Do you want to print a progress bar out to the console? Defaults to TRUE.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments, currently unused</td>
</tr>
</tbody>
</table>

Details

refit allows you to (a) update the probabilities (i.e. weights) of a previously-fit model with new data or additional iterations and (b) optionally use beta of a previously-fit LDA topic model as the eta prior for the new model. This is tuned by setting beta_as_prior = FALSE or beta_as_prior = TRUE respectively.

prior_weight tunes how strong the base model is represented in the prior. If prior_weight = 1, then the tokens from the base model’s training data have the same relative weight as tokens in new_data. In other words, it is like just adding training data. If prior_weight is less than 1, then tokens in new_data are given more weight. If prior_weight is greater than 1, then the tokens from the base model’s training data are given more weight.

If prior_weight is NA, then the new eta is equal to eta from the old model, with new tokens folded in. (For handling of new tokens, see below.) Effectively, this just controls how the sampler initializes (described below), but does not give prior weight to the base model.

Instead of initializing token-topic assignments in the manner for new models (see tidylda), the update initializes in 2 steps:

First, topic-document probabilities (i.e. theta) are obtained by a call to predict.tidylda using method = "dot" for the documents in new_data. Next, both beta and theta are passed to an internal function, initialize_topic_counts, which assigns topics to tokens in a manner approximately proportional to the posteriors and executes a single Gibbs iteration.

refit handles the addition of new vocabulary by adding a flat prior over new tokens. Specifically, each entry in the new prior is equal to the 10th percentile of eta from the old model. The resulting model will have the total vocabulary of the old model plus any new vocabulary tokens. In other words, after running refit.tidylda ncol(beta) >= ncol(new_data) where beta is from the new model and new_data is the additional data.
You can add additional topics by setting the `additional_k` parameter to an integer greater than zero. New entries to `alpha` have a flat prior equal to the median value of `alpha` in the old model. (Note that if `alpha` itself is a flat prior, i.e. scalar, then the new topics have the same value for their prior.) New entries to `eta` are the average of all previous topics in `eta`.

**Value**

Returns an S3 object of class c("tidylda").

**Note**

Updates are, as of this writing, are almost-surely useful but their behaviors have not been optimized or well-studied. Caveat emptor!

**Examples**

```r
# load a document term matrix
data(nih_sample_dtm, package = "textmineR")

d1 <- nih_sample_dtm[1:50, ]
d2 <- nih_sample_dtm[51:100, ]

# fit a model
m <- tidylda(d1,
  k = 10,
  iterations = 200, burnin = 175
)

# update an existing model by adding documents using old model as prior
m2 <- refit(
  object = m,
  new_data = rbind(d1, d2),
  iterations = 200,
  burnin = 175,
  prior_weight = 1
)

# use an old model to initialize new model and not use old model as prior
m3 <- refit(
  object = m,
  new_data = d2, # new documents only
  iterations = 200,
  burnin = 175,
  prior_weight = NA
)

# add topics while updating a model by adding documents
m4 <- refit(
  object = m,
  new_data = rbind(d1, d2),
```
tidy.tidylda

**Tidy a matrix from a tidylda topic model**

### Description
Tidy the result of a tidylda topic model

### Usage

```r
## S3 method for class 'tidylda'
tidy(x, matrix, log = FALSE, ...)
## S3 method for class 'matrix'
tidy(x, matrix, log = FALSE, ...)
```

### Arguments

- **x**: an object of class tidylda or an individual beta, theta, or lambda matrix.
- **matrix**: the matrix to tidy; one of 'beta', 'theta', or 'lambda'
- **log**: do you want to have the result on a log scale? Defaults to FALSE
- **...**: other arguments passed to methods, currently not used

### Value
Returns a tibble.

If `matrix = "beta"` then the result is a table of one row per topic and token with the following columns: topic, token, beta

If `matrix = "theta"` then the result is a table of one row per document and topic with the following columns: document, topic, theta

If `matrix = "lambda"` then the result is a table of one row per topic and token with the following columns: topic, token, lambda

### Functions

- `tidy.matrix`: Tidy an individual matrix. Useful for predictions and called from tidy.tidylda

### Note
If `log = TRUE` then "log_" will be appended to the name of the third column of the resulting table. e.g. "beta" becomes "log_beta".
tidylda

Examples

dtm <- textmineR::nih_sample_dtm

lda <- tidylda(data = dtm, k = 10, iterations = 100, burnin = 75)

tidy_beta <- tidy(lda, matrix = "beta")

tidy_theta <- tidy(lda, matrix = "theta")

tidy_lambda <- tidy(lda, matrix = "lambda")

tidylda

Fit a Latent Dirichlet Allocation topic model

Description

Fit a Latent Dirichlet Allocation topic model using collapsed Gibbs sampling.

Usage

tidylda(
  data,
  k,
  iterations = NULL,
  burnin = -1,
  alpha = 0.1,
  eta = 0.05,
  optimize_alpha = FALSE,
  calc_likelihood = TRUE,
  calc_r2 = FALSE,
  threads = 1,
  return_data = FALSE,
  verbose = TRUE,
  ...
)

Arguments

data A document term matrix or term co-occurrence matrix. The preferred class is a
dgCMMatrix-class. However there is support for any Matrix-class object as
well as several other commonly-used classes such as matrix, dfm, DocumentTermMatrix,
and simple_triplet_matrix

k Integer number of topics.

iterations Integer number of iterations for the Gibbs sampler to run.
tidylda

burnin Integer number of burnin iterations. If burnin is greater than -1, the resulting "beta" and "theta" matrices are an average over all iterations greater than burnin.

alpha Numeric scalar or vector of length k. This is the prior for topics over documents.

eta Numeric scalar, numeric vector of length ncol(data), or numeric matrix with k rows and ncol(data) columns. This is the prior for words over topics.

optimize_alpha Logical. Do you want to optimize alpha every iteration? Defaults to FALSE. See 'details' below for more information.

calc_likelihood Logical. Do you want to calculate the log likelihood every iteration? Useful for assessing convergence. Defaults to TRUE.

calc_r2 Logical. Do you want to calculate R-squared after the model is trained? Defaults to FALSE. This calls CalcTopicModelR2.

threads Number of parallel threads, defaults to 1. See Details, below.

return_data Logical. Do you want data returned as part of the model object?

verbose Logical. Do you want to print a progress bar out to the console? Defaults to TRUE.

... Additional arguments, currently unused

Details

This function calls a collapsed Gibbs sampler for Latent Dirichlet Allocation written using the excellent Rcpp package. Some implementation notes follow:

Topic-token and topic-document assignments are not initialized based on a uniform-random sampling, as is common. Instead, topic-token probabilities (i.e. beta) are initialized by sampling from a Dirichlet distribution with eta as its parameter. The same is done for topic-document probabilities (i.e. theta) using alpha. Then an internal function is called (initialize_topic_counts) to run a single Gibbs iteration to initialize assignments of tokens to topics and topics to documents.

When you use burn-in iterations (i.e. burnin = TRUE), the resulting beta and theta matrices are calculated by averaging over every iteration after the specified number of burn-in iterations. If you do not use burn-in iterations, then the matrices are calculated from the last run only. Ideally, you’d burn in every iteration before convergence, then average over the chain after its converged (and thus every observation is independent).

If you set optimize_alpha to TRUE, then each element of alpha is proportional to the number of times each topic has been sampled that iteration averaged with the value of alpha from the previous iteration. This lets you start with a symmetric alpha and drift into an asymmetric one. However, (a) this probably means that convergence will take longer to happen or convergence may not happen at all. And (b) I make no guarantees that doing this will give you any benefit or that it won’t hurt your model. Caveat emptor!

The log likelihood calculation is the same that can be found on page 9 of https://arxiv.org/pdf/1510.08628.pdf. The only difference is that the version in tidylda allows eta to be a vector or matrix. (Vector used in this function, matrix used for model updates in refit.tidylda. At present, the log likelihood function appears to be ok for assessing convergence. i.e. It has the right shape. However, it is, as of this writing, returning positive numbers, rather than the expected negative numbers. Looking into that, but in the meantime caveat emptor once again.
Parallelism is not currently implemented. The `threads` argument is a placeholder for planned enhancements.

**Value**

Returns an S3 object of class `tidylda`. See `new_tidyllda`.

**Examples**

```r
# load some data
data(nih_sample_dtm, package = "textmineR")

# fit a model
set.seed(12345)
m <- tidylda(
  data = nih_sample_dtm[1:20,], k = 5,
  iterations = 200, burnin = 175
)

str(m)

# predict on held-out documents using gibbs sampling "fold in"
p1 <- predict(m, nih_sample_dtm[21:100,],
  method = "gibbs",
  iterations = 200, burnin = 175
)

# predict on held-out documents using the dot product method
p2 <- predict(m, nih_sample_dtm[21:100,], method = "dot")

# compare the methods
barplot(rbind(p1[1,], p2[1,]), beside = TRUE, col = c("red", "blue"))
```
Index

augment.tidylda, 2

calc_prob_coherence, 3
CalcTopicModelR2, 10, 14

dfm, 3, 13
DocumentTermMatrix, 3, 13

FitLdaModel, 10

generate.tidylda_posterior (posterior), 5
glance.tidylda, 4

initialize_topic_counts, 10, 14

matrix, 3, 13

new_tidylda, 15

posterior, 5

predict.tidylda, 6, 10

print.tidylda, 8

refit.tidylda, 9, 14

simple_triplet_matrix, 3, 13

tdm_tidiers, 2
tibble, 4, 8, 12
tidy.matrix (tidy.tidylda), 12
tidy.tidylda, 12
tidylda, 2, 7, 10, 13, 14