Package ‘lda’

November 3, 2011

Type Package

Title Collapsed Gibbs sampling methods for topic models.

Version 1.3.1

Date 2011-11-02

Author Jonathan Chang

Maintainer Jonathan Chang <`jonchang@fb.com`>, Andrew Dai <`a.dai@ed.ac.uk`>

Description This package implements latent Dirichlet allocation (LDA)
and related models. This includes (but is not limited to)
sLDA, corrLDA, and the mixed-membership stochastic blockmodel.
Inference for all of these models is implemented via a fast
collapsed Gibbs sampler writtten in C. Utility functions for
reading/writing data typically used in topic models, as well as
tools for examining posterior distributions are also included.

License LGPL

LazyLoad yes

Suggests Matrix, ggplot2, penalized

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2011-11-03 11:48:11

R topics documented:

`lda-package` *Collapsed Gibbs Samplers and Related Utility Functions for LDA-type Models*

Description

This package contains functions to read in text corpora, fit LDA-type models to them, and use the fitted models to explore the data and make predictions.

Details

Package:	lda
Type:	Package
Version:	1.3
Date:	2011-10-26
License:	BSD
LazyLoad:	yes

Author(s)

Jonathan Chang (<`jonchang@fb.com`>) Andrew Dai (<`a.dai@ed.ac.uk`>)

Special thanks to the following for their reports and comments: Edo Airoldi, Jordan Boyd-Graber, Christopher E. Cramer, James Danowski, Khalid El-Arini, Roger Levy, Solomon Messing, Joerg Reichardt

References

Blei, David M. and Ng, Andrew and Jordan, Michael. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

See Also

Functions to fit models: `lda.collapsed.gibbs.sampler` `slda.em` `mmsb.collapsed.gibbs.sampler` `nubbi.collapsed.gibbs.sampler` `rtm.collapsed.gibbs.sampler`

Functions to read/create corpora: `lexicalize` `read.documents` `read.vocab`

Functions to manipulate corpora: `merge.documents` `filter.words` `shift.word.indices` `links.as.edgelist`

Functions to compute summary statistics on corpora: `word.counts` `document.lengths`

Functions which use the output of fitted models: `predictive.distribution` `top.topic.words` `top.topic.documents` `predictive.link.probability`

Included data sets: `cora` `poliblog` `sampson`

Examples

```
## See demos for the following three common use cases:

## Not run: demo(lda)

## Not run: demo(slda)

## Not run: demo(mmsb)

## Not run: demo(rtm)
```

`cora` *A subset of the Cora dataset of scientific documents.*

Description

A collection of 2410 scientific documents in LDA format with links and titles from the Cora search engine.

Usage

```
data(cora.documents)
data(cora.vocab)
data(cora.cites)
data(cora.titles)
```

Format

`cora.documents` and `cora.vocab` comprise a corpus of 2410 documents conforming to the LDA format.

`cora.titles` is a character vector of titles for each document (i.e., each entry of `cora.documents`).

`cora.cites` is a list representing the citations between the documents in the collection (see related for format).

Source

Automating the construction of internet protals with machine learning. McCallum et al. Information Retrieval. 2000.

See Also

`lda.collapsed.gibbs.sampler` for the format of the corpus.

`rtm.collapsed.gibbs.sampler` for the format of the citation links.

Examples

```
data(cora.documents)
data(cora.vocab)
data(cora.links)
data(cora.titles)
```

`filter.words` *Functions to manipulate text corpora in LDA format.*

Description

`merge.documents` concatenates a set of documents. `filter.words` removes references to certain words from a collection of documents. `shift.word.indices` adjusts references to words by a fixed amount.

Usage

```
merge.documents(...)
filter.words(documents, to.remove)
shift.word.indices(documents, amount)
```

Arguments

`...`	For `merge.documents`, the set of corpora to be merged. All arguments to `...` must be corpora of the same length. The documents in the same position in each of the arguments will be concatenated, i.e., the new document 1 will be the concatenation of document 1 from argument 1, document 2 from argument 1, etc.
`documents`	For `filter.words` and `shift.word.indices`, the corpus to be operated on.
`to.remove`	For `filter.words`, an integer vector of words to filter. The words in each document which also exist in `to.remove` will be removed.
`amount`	For `shift.word.indices`, an integer scalar by which to shift the vocabulary in the corpus. `amount` will be added to each entry of the word field in the corpus.

Value

A corpus with the documents merged/words filtered/words shifted. The format of the input and output corpora is described in `lda.collapsed.gibbs.sampler`.

Author(s)

Jonathan Chang (<`jonchang@fb.com`>)

See Also

`lda.collapsed.gibbs.sampler` for the format of the return value.

`word.counts` to compute statistics associated with a corpus.

Examples

```
data(cora.documents)

## Just use a small subset for the example.
corpus <- cora.documents[1:6]
## Get the word counts.
wc <- word.counts(corpus)

## Only keep the words which occur more than 4 times.
filtered <- filter.words(corpus,
                         as.numeric(names(wc)[wc <= 4]))
## [[1]]
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1   23   34   37   44
## [2,]    4    1    3    4    1
##
## [[2]]
##      [,1] [,2]
## [1,]   34   94
## [2,]    1    1
## ... long output ommitted ...

## Shift the second half of the corpus.
shifted <- shift.word.indices(filtered[4:6], 100)
## [[1]]
##      [,1] [,2] [,3]
## [1,]  134  281  307
## [2,]    2    5    7
##
## [[2]]
##      [,1] [,2]
## [1,]  101  123
## [2,]    1    4
##
## [[3]]
##      [,1] [,2]
## [1,]  101  194
## [2,]    6    3

## Combine the unshifted documents and the shifted documents.
merge.documents(filtered[1:3], shifted)
## [[1]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1   23   34   37   44  134  281  307
## [2,]    4    1    3    4    1    2    5    7
##
## [[2]]
##      [,1] [,2] [,3] [,4]
## [1,]   34   94  101  123
## [2,]    1    1    1    4
##
## [[3]]
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   34   37   44   94  101  194
## [2,]    4    1    7    1    6    3
```

`lda.collapsed.gibbs.sampler`
Functions to Fit LDA-type models

Description

These functions use a collapsed Gibbs sampler to fit three different models: latent Dirichlet allocation (LDA), the mixed-membership stochastic blockmodel (MMSB), and supervised LDA (sLDA). These functions take sparsely represented input documents, perform inference, and return point estimates of the latent parameters using the state at the last iteration of Gibbs sampling.

Usage

```
lda.collapsed.gibbs.sampler(documents, K, vocab, num.iterations, alpha,
eta, initial = NULL, burnin = NULL, compute.log.likelihood = FALSE,
  trace = 0L, freeze.topics = FALSE)

slda.em(documents, K, vocab, num.e.iterations, num.m.iterations, alpha,
eta, annotations, params, variance, logistic = FALSE, lambda = 10,
regularise = FALSE, method = "sLDA", trace = 0L)

mmsb.collapsed.gibbs.sampler(network, K, num.iterations, alpha,
beta.prior, initial = NULL, burnin = NULL, trace = 0L)
```

Arguments

`documents`	A list whose length is equal to the number of documents, D. Each element of *documents* is an integer matrix with two rows. Each column of *documents[[i]]* (i.e., document i) represents a word occurring in the document.

	documents[[i]][1, j] is a 0-indexed word identifier for the jth word in document i. That is, this should be an index - 1 into *vocab*. *documents[[i]][2, j]* is an integer specifying the number of times that word appears in the document.
`network`	For `mmsb.collapsed.gibbs.sampler`, a $D \times D$ matrix (coercible as logical) representing the adjacency matrix for the network. Note that elements on the diagonal are ignored.
`K`	An integer representing the number of topics in the model.
`vocab`	A character vector specifying the vocabulary words associated with the word indices used in *documents*.
`num.iterations`	The number of sweeps of Gibbs sampling over the entire corpus to make.
`num.e.iterations`	For `slda.em`, the number of Gibbs sampling sweeps to make over the entire corpus for each iteration of EM.
`num.m.iterations`	For `slda.em`, the number of EM iterations to make.
`alpha`	The scalar value of the Dirichlet hyperparameter for topic proportions.
`beta.prior`	For `mmsb.collapsed.gibbs.sampler`, the the beta hyperparameter for each entry of the block relations matrix. This parameter should be a length-2 list whose entries are $K \times K$ matrices. The elements of the two matrices comprise the two parameters for each beta variable.
`eta`	The scalar value of the Dirichlet hyperparamater for topic multinomials.
`initial`	A list of initial topic assignments for words. It should be in the same format as the *assignments* field of the return value. If this field is NULL, then the sampler will be initialized with random assignments.
`burnin`	A scalar integer indicating the number of Gibbs sweeps to consider as burn-in (i.e., throw away) for `lda.collapsed.gibbs.sampler` and `mmsb.collapsed.gibbs.sampl` If this parameter is non-NULL, it will also have the side-effect of enabling the *document_expects* field of the return value (see below for details). Note that burnin iterations do NOT count towards *num.iterations*.
`compute.log.likelihood`	A scalar logical which when `TRUE` will cause the sampler to compute the log likelihood of the words (to within a constant factor) after each sweep over the variables. The log likelihood for each iteration is stored in the *log.likelihood* field of the result. This is useful for assessing convergence, but slows things down a tiny bit.
`annotations`	A length D numeric vector of covariates associated with each document. Only used by `slda.em` which models documents along with numeric annotations associated with each document.
`params`	For `slda.em`, a length K numeric vector of regression coefficients at which the EM algorithm should be initialized.
`variance`	For `slda.em`, the variance associated with the Gaussian response modeling the annotations in *annotations*.
`logistic`	For `slda.em`, a scalar logical which, when `TRUE`, causes the annotations to be modeled using a logistic response instead of a Gaussian (the covariates will be coerced as logicals).

`lambda`	When *regularise* is `TRUE`. This is a scalar that is the standard deviation of the Gaussian prior on the regression coefficients.
`regularise`	When `TRUE`, a Gaussian prior is used for the regression coefficients. This requires the `penalized` package.
`method`	For `slda.em`, a character indicating how to model the annotations. Only `"sLDA"`, the stock model given in the references, is officially supported at the moment.
`trace`	When `trace` is greater than zero, diagnostic messages will be output. Larger values of `trace` imply more messages.
`freeze.topics`	When `TRUE`, topic assignments will occur but the counts of words associated with topics will not change. *initial* should be set when this option is used. This is best use for sampling test documents.

Value

A fitted model as a list with the following components:

`assignments`	A list of length D. Each element of the list, say `assignments[[i]]` is an integer vector of the same length as the number of columns in `documents[[i]]` indicating the topic assignment for each word.
`topics`	A $K \times V$ matrix where each entry indicates the number of times a word (column) was assigned to a topic (row). The column names should correspond to the vocabulary words given in *vocab*.
`topic_sums`	A length K vector where each entry indicates the total number of times words were assigned to each topic.
`document_sums`	A $K \times D$ matrix where each entry is an integer indicating the number of times words in each document (column) were assigned to each topic (column).
`log.likelihoods`	Only for `lda.collapsed.gibbs.sampler`. A matrix with 2 rows and `num.iterations` columns of log likelihoods when the flag `compute.log.likelihood` is set to `TRUE`. The first row contains the full log likelihood (including the prior), whereas the second row contains the log likelihood of the observations conditioned on the assignments.
`document_expects`	This field only exists if *burnin* is non-NULL. This field is like document_sums but instead of only aggregating counts for the last iteration, this field aggegates counts over all iterations after burnin.
`net.assignments.left`	Only for `mmsb.collapsed.gibbs.sampler`. A $D \times D$ integer matrix of topic assignments for the source document corresponding to the link between one document (row) and another (column).
`net.assignments.right`	Only for `mmsb.collapsed.gibbs.sampler`. A $D \times D$ integer matrix of topic assignments for the destination document corresponding to the link between one document (row) and another (column).

`blocks.neg`	Only for `mmsb.collapsed.gibbs.sampler`. A $K \times K$ integer matrix indicating the number of times the source of a non-link was assigned to a topic (row) and the destination was assigned to another (column).
`blocks.pos`	Only for `mmsb.collapsed.gibbs.sampler`. A $K \times K$ integer matrix indicating the number of times the source of a link was assigned to a topic (row) and the destination was assigned to another (column).
`model`	For `slda.em`, a model of type `lm`, the regression model fitted to the annotations.
`coefs`	For `slda.em`, a length K numeric vector of coefficients for the regression model.

Note

WARNING: This function does not compute precisely the correct thing when the count associated with a word in a document is not 1 (this is for speed reasons currently). A workaround when a word appears multiple times is to replicate the word across several columns of a document. This will likely be fixed in a future version.

Author(s)

Jonathan Chang (`<jonchang@fb.com>`)

References

Blei, David M. and Ng, Andrew and Jordan, Michael. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

Airoldi , Edoardo M. and Blei, David M. and Fienberg, Stephen E. and Xing, Eric P. Mixed Membership Stochastic Blockmodels. Journal of Machine Learning Research, 2008.

Blei, David M. and McAuliffe, John. Supervised topic models. Advances in Neural Information Processing Systems, 2008.

Griffiths, Thomas L. and Steyvers, Mark. Finding scientific topics. Proceedings of the National Academy of Sciences, 2004.

See Also

`read.documents` and `lexicalize` can be used to generate the input data to these models.

`top.topic.words`, `predictive.distribution`, and `slda.predict` for operations on the fitted models.

Examples

```
## See demos for the three functions:

## Not run: demo(lda)

## Not run: demo(slda)

## Not run: demo(mmsb)
```

`lexicalize` *Generate LDA Documents from Raw Text*

Description

This function reads raw text in *doclines* format and returns a corpus and vocabulary suitable for the inference procedures defined in the **lda** package.

Usage

```
lexicalize(doclines, sep = " ", lower = TRUE, count = 1L, vocab = NULL)
```

Arguments

`doclines`	A character vector of document lines to be used to construct a corpus. See details for a description of the format of these lines.
`sep`	Separator string which is used to tokenize the input strings (default '` `').
`lower`	Logical indicating whether or not to convert all tokens to lowercase (default '`TRUE`').
`count`	An integer scaling factor to be applied to feature counts. A single observation of a feature will be rendered as *count* observations in the return value (the default value, '`1`', is appropriate in most cases).
`vocab`	If left unspecified (or `NULL`), the vocabulary for the corpus will be automatically inferred from the observed tokens. Otherwise, this parameter should be a character vector specifying acceptable tokens. Tokens not appearing in this list will be filtered from the documents.

Details

This function first tokenizes a character vector by splitting each entry of the vector by *sep* (note that this is currently a fixed separator, not a regular expression). If *lower* is '`TRUE`', then the tokens are then all converted to lowercase.

At this point, if *vocab* is `NULL`, then a vocabulary is constructed from the set of unique tokens appearing across all character vectors. Otherwise, the tokens derived from the character vectors are filtered so that only those appearing in *vocab* are retained.

Finally, token instances within each document (i.e., original character string) are tabulated in the format described in `lda.collapsed.gibbs.sampler`.

Value

If *vocab* is unspecified or `NULL`, a list with two components:

`documents`	A list of document matrices in the format described in `lda.collapsed.gibbs.sampler`.
`vocab`	A character vector of unique tokens occurring in the corpus.

Note

Because of the limited tokenization and filtering capabilities of this function, it may not be useful in many cases. This may be resolved in a future release.

Author(s)

Jonathan Chang (`<jonchang@fb.com>`)

See Also

`lda.collapsed.gibbs.sampler` for the format of the return value.

`read.documents` to generate the same output from a file encoded in LDA-C format.

`word.counts` to compute statistics associated with a corpus.

`merge.documents` for operations on a collection of documents.

Examples

```
## Generate an example.
example <- c("I am the very model of a modern major general",
             "I have a major headache")

corpus <- lexicalize(example, lower=TRUE)

## corpus$documents:
## $documents[[1]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    0    1    2    3    4    5    6    7    8     9
## [2,]    1    1    1    1    1    1    1    1    1     1
##
## $documents[[2]]
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0   10    6    8   11
## [2,]    1    1    1    1    1

## corpus$lexicon:
## $vocab
## [1] "i"        "am"       "the"      "very"     "model"    "of"
## [7] "a"        "modern"   "major"    "general"  "have"     "headache"

## Only keep words that appear at least twice:
to.keep <- corpus$vocab[word.counts(corpus$documents, corpus$vocab) >= 2]

## Re-lexicalize, using this subsetted vocabulary
documents <- lexicalize(example, lower=TRUE, vocab=to.keep)

## documents:
## [[1]]
##      [,1] [,2] [,3]
## [1,]    0    1    2
## [2,]    1    1    1
```

```
##
## [[2]]
##      [,1] [,2] [,3]
## [1,]    0    1    2
## [2,]    1    1    1
```

links.as.edgelist *Convert a set of links keyed on source to a single list of edges.*

Description

This function takes as input a collection of links (as used/described by the model fitting functions in this package) and reproduces the links as a matrix.

Usage

```
links.as.edgelist(links)
```

Arguments

`links`	A list of links; the format of this is described in `rtm.collapsed.gibbs.sampler`.

Value

A two-column matrix where each row represents an edge. Note that the indices in this matrix are 1-indexed rather than 0-indexed.

Author(s)

Jonathan Chang (<`jonchang@fb.com`>)

See Also

`rtm.collapsed.gibbs.sampler` for the input format. `predictive.link.probability` is a usage example of the output of this function.

Examples

```
## Take the citations for the first few documents of Cora.
data(cora.cites)

links <- cora.cites[1:5]
links
## [[1]]
## [1] 484 389

## [[2]]
## integer(0)
## [[3]]
## integer(0)

## [[4]]
## [1] 177 416 533

## [[5]]
## [1] 153

links.as.edgelist(links)
##      [,1] [,2]
## [1,]    1  485
## [2,]    1  390
## [3,]    4  178
## [4,]    4  417
## [5,]    4  534
## [6,]    5  154
```

`nubbi.collapsed.gibbs.sampler` *Collapsed Gibbs Sampling for the Networks Uncovered By Bayesian Inference (NUBBI) Model.*

Description

Fit a NUBBI model, which takes as input a collection of entities with corresponding textual descriptions as well as a set of descriptions for pairs of entities. The NUBBI model the produces a latent space description of both the entities and the relationships between them.

Usage

```
nubbi.collapsed.gibbs.sampler(contexts, pair.contexts, pairs, K.individual, K.pair,
```

Arguments

`contexts`	The set of textual descriptions (i.e., documents) for individual entities in LDA format (see `lda.collapsed.gibbs.sampler` for details).
`pair.contexts`	A set of textual descriptions for pairs of entities, also in LDA format.
`pairs`	Labelings as to which pair each element of `pair.contexts` refer to. This parameter should be an integer matrix with two columns and the same number of rows as `pair.contexts`. The two elements in each row of `pairs` are 0-indexed indices into `contexts` indicating which two entities that element of `pair.contexts` describes.
`K.individual`	A scalar integer representing the number of topics for the individual entities.
`K.pair`	A scalar integer representing the number of topics for entity pairs.

`vocab`	A character vector specifying the vocabulary words associated with the word indices used in *contexts* and *pair.contexts*.
`num.iterations`	The number of sweeps of Gibbs sampling over the entire corpus to make.
`alpha`	The scalar value of the Dirichlet hyperparameter for topic proportions.
`eta`	The scalar value of the Dirichlet hyperparamater for topic multinomials.
`xi`	The scalar value of the Dirichlet hyperparamater for source proportions.

Details

The NUBBI model is a switching model wherein the description of each entity-pair can be ascribed to either the first entity of the pair, the second entity of the pair, or their relationship. The NUBBI model posits a latent space (i.e., topic model) over the individual entities, and a different latent space over entity relationships.

The collapsed Gibbs sampler used in this model is different than the variational inference method proposed in the paper and is highly experimental.

Value

A fitted model as a list with the same components as returned by `lda.collapsed.gibbs.sampler` with the following additional components:

`source_assignments`	A list of `length(pair.contexts)` whose elements `source_assignments[[i]]` are of the same length as `pair.contexts[[i]]` where each entry is either 0 if the sampler assigned the word to the first entity, 1 if the sampler assigned the word to the second entity, or 2 if the sampler assigned the word to the relationship between the two.
x	
`document_source_sums`	A matrix with three columns and `length(pair.contexts)` rows where each row indicates how many words were assigned to the first entity of the pair, the second entity of the pair, and the relationship between the two, respectively.
`document_sums`	Semantically similar to the entry in `lda.collapsed.gibbs.sampler`, except that it is a list whose first `length(contexts)` correspond to the columns of the entry in `lda.collapsed.gibbs.sampler` for the individual contexts, and the remaining `length(pair.contexts)` entries correspond to the columns for the pair contexts.
`topics`	Like the entry in `lda.collapsed.gibbs.sampler`, except that it contains the concatenation of the `K.individual` topics and the `K.pair` topics.

Note

The underlying sampler is quite general and could potentially be used for other models such as the author-topic model (McCallum et al.) and the citation influence model (Dietz et al.). Please examine the source code and/or contact the author(s) for further details.

Author(s)

Jonathan Chang (<jonchang@fb.com>)

References

Chang, Jonathan and Boyd-Graber, Jordan and Blei, David M. Connections between the lines: Augmenting social networks with text. KDD, 2009.

See Also

See `lda.collapsed.gibbs.sampler` for a description of the input formats and similar models.

`rtm.collapsed.gibbs.sampler` is a different kind of model for document networks.

Examples

```
## See demo.

## Not run: demo(nubbi)
```

`poliblog` *A collection of political blogs with ratings.*

Description

A collection of 773 political blogs in LDA format with conservative/liberal ratings.

Usage

```
data(poliblog.documents)
data(poliblog.vocab)
data(poliblog.ratings)
```

Format

`poliblog.documents` and `poliblog.vocab` comprise a corpus of 773 political blogs conforming to the LDA format.

`poliblog.ratings` is a numeric vector of length 773 which gives a rating of liberal (-100) or conservative (100) to each document in the corpus.

Source

Blei, David M. and McAuliffe, John. Supervised topic models. Advances in Neural Information Processing Systems, 2008.

See Also

`lda.collapsed.gibbs.sampler` for the format of the corpus.

Examples

```
data(poliblog.documents)
data(poliblog.vocab)
data(poliblog.ratings)
```

`predictive.distribution` *Compute predictive distributions for fitted LDA-type models.*

Description

This function takes a fitted LDA-type model and computes a predictive distribution for new words in a document. This is useful for making predictions about held-out words.

Usage

```
predictive.distribution(document_sums, topics, alpha, eta)
```

Arguments

`document_sums`	A $K \times D$ matrix where each entry is a numeric proportional to the probability of seeing a topic (row) conditioned on document (column) (this entry is sometimes denoted $\theta_{d,k}$ in the literature, see details). Either the *document_sums* field or the *document_expects* field from the output of `lda.collapsed.gibbs.sampler` can be used.
`topics`	A $K \times V$ matrix where each entry is a numeric proportional to the probability of seeing the word (column) conditioned on topic (row) (this entry is sometimes denoted $\beta_{w,k}$ in the literature, see details). The column names should correspond to the words in the vocabulary. The *topics* field from the output of `lda.collapsed.gibbs.sampler` can be used.
`alpha`	The scalar value of the Dirichlet hyperparameter for topic proportions. See references for details.
`eta`	The scalar value of the Dirichlet hyperparamater for topic multinomials. See references for details.

Details

The formula used to compute predictive probability is $p_d(w) = \sum_k (\theta_{d,k} + \alpha)(\beta_{w,k} + \eta)$.

Value

A $V \times D$ matrix of the probability of seeing a word (row) in a document (column). The row names of the matrix are set to the column names of *topics*.

Author(s)

Jonathan Chang (<jonchang@fb.com>)

References

Blei, David M. and Ng, Andrew and Jordan, Michael. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

See Also

`lda.collapsed.gibbs.sampler` for the format of *topics* and *document_sums* and details of the model.

`top.topic.words` demonstrates another use for a fitted topic matrix.

Examples

```
## Fit a model (from demo(lda)).
data(cora.documents)
data(cora.vocab)

K <- 10 ## Num clusters
result <- lda.collapsed.gibbs.sampler(cora.documents,
                                      K,  ## Num clusters
                                      cora.vocab,
                                      25,  ## Num iterations
                                      0.1,
                                      0.1)

## Predict new words for the first two documents
predictions <-  predictive.distribution(result$document_sums[,1:2],
                                        result$topics,
                                        0.1, 0.1)

## Use top.topic.words to show the top 5 predictions in each document.
top.topic.words(t(predictions), 5)

##      [,1]         [,2]
## [1,] "learning"   "learning"
## [2,] "algorithm"  "paper"
## [3,] "model"      "problem"
## [4,] "paper"      "results"
## [5,] "algorithms" "system"
```

`predictive.link.probability`

Use the RTM to predict whether a link exists between two documents.

Description

This function takes a fitted LDA-type model (e.g., LDA or RTM) and makes predictions about the likelihood of a link existing between pairs of documents.

Usage

```
predictive.link.probability(edgelist, document_sums, alpha, beta)
```

Arguments

`edgelist`	A two-column integer matrix where each row represents an edge on which to make a prediction. An edge is expressed as a pair of integer indices (1-indexed) into the columns (i.e., documents) of *document_sums* (see below).
`document_sums`	A $K \times D$ matrix where each entry is a numeric proportional to the probability of seeing a topic (row) conditioned on document (column) (this entry is sometimes denoted $\theta_{d,k}$ in the literature, see details). The *document_sums* field or the *document_expects* field from the output of `lda.collapsed.gibbs.sampler` and `rtm.collapsed.gibbs.sampler` can be used.
`alpha`	The value of the Dirichlet hyperparamter generating the distribution over *document_sums*. This, in effect, smooths the similarity between documents.
`beta`	A numeric vector of regression weights which is used to determine the similarity between two vectors (see details). Arguments will be recycled to create a vector of length `dim(document_sums)[1]`.

Details

Whether or not a link exists between two documents i and j is a function of the weighted inner product of the `document_sums[,i]` and `document_sums[,j]`. After normalizing `document_sums` column-wise, this inner product is weighted by *beta*.

This quantity is then passed to a link probability function. Like `rtm.collapsed.gibbs.sampler` in this package, only the exponential link probability function is supported. Note that quantities are automatically scaled to be between 0 and 1.

Value

A numeric vector of length `dim(edgelist)[1]`, representing the probability of a link existing between each pair of documents given in the edge list.

Author(s)

Jonathan Chang (`<jonchang@fb.com>`)

References

Chang, Jonathan and Blei, David M. Relational Topic Models for Document Networks. Artificial intelligence and statistics. 2009.

See Also

`rtm.collapsed.gibbs.sampler` for the format of *document_sums*. `links.as.edgelist` produces values for *edgelist*. `predictive.distribution` makes predictions about document content instead.

Examples

```
## See demo.

## Not run: demo(rtm)
```

`read.documents` *Read LDA-formatted Document and Vocabulary Files*

Description

These functions read in the document and vocabulary files associated with a corpus. The format of the files is the same as that used by LDA-C (see below for details). The return value of these functions can be used by the inference procedures defined in the **lda** package.

Usage

```
read.documents(filename = "mult.dat")

read.vocab(filename = "vocab.dat")
```

Arguments

`filename`	A length-1 character vector specifying the path to the document/vocabulary file. These are set to '`mult.dat`' and '`vocab.dat`' by default.

Details

The details of the format are also described in the readme for LDA-C.

The format of the documents file is appropriate for typical text data as it sparsely encodes observed features. A single file encodes a *corpus* (a collection of documents). Each line of the file encodes a single *document* (a feature vector).

The line encoding a document begins with an integer followed by a number of *feature-count pairs*, all separated by spaces. A feature-count pair consists of two integers separated by a colon. The first integer indicates the feature (note that this is zero-indexed!) and the second integer indicates the count (i.e., value) of that feature. The initial integer of a line indicates how many feature-count pairs are to be expected on that line.

Note that we permit a feature to appear more than once on a line, in which case the value for that feature will be the sum of all instances (the behavior for such files is undefined for LDA-C). For example, a line reading '`4 7:1 0:2 7:3 1:1`' will yield a document with feature 0 occurring twice, feature 1 occurring once, and feature 7 occurring four times, with all other features occurring zero times.

The format of the vocabulary is a set of newline separated strings corresponding to features. That is, the first line of the vocabulary file will correspond to the label for feature 0, the second for feature 1, etc.

Value

`read.documents` returns a list of matrices suitable as input for the inference routines in **lda**. See `lda.collapsed.gibbs.sampler` for details.

`read.vocab` returns a character vector of strings corresponding to features.

Author(s)

Jonathan Chang (`<jonchang@fb.com>`)

References

Blei, David M. Latent Dirichlet Allocation in C. *`http://www.cs.princeton.edu/~blei/lda-c/index.html`*

See Also

`lda.collapsed.gibbs.sampler` for the format of the return value of `read.documents`.

`lexicalize` to generate the same output from raw text data.

`word.counts` to compute statistics associated with a corpus.

`merge.documents` for operations on a collection of documents.

Examples

```
## Read files using default values.
## Not run: setwd("corpus directory")
## Not run: documents <- read.documents()
## Not run: vocab <- read.vocab()

## Read files from another location.
## Not run: documents <- read.documents("corpus directory/features")
## Not run: vocab <- read.vocab("corpus directory/labels")
```

`rtm.collapsed.gibbs.sampler` *Collapsed Gibbs Sampling for the Relational Topic Model (RTM).*

Description

Fit a generative topic model which accounts for both the words which occur in a collection of documents as well as the links between the documents.

Usage

```
rtm.collapsed.gibbs.sampler(documents, links, K, vocab, num.iterations,
  alpha, eta, beta, trace = 0L, test.start = length(documents) + 1L)
rtm.em(documents, links, K, vocab, num.e.iterations, num.m.iterations,
        alpha, eta, lambda = sum(sapply(links, length))/(length(links) * (length(li
  initial.beta = rep(3, K), trace = 0L,
  test.start = length(documents) + 1L)
```

Arguments

`documents`	A collection of documents in LDA format. See `lda.collapsed.gibbs.sampler` for details.
`links`	A list representing the connections between the documents. This list should be of the same length as the *documents*. Each element, `links[[i]]`, is an integer vector expressing connections between document *i* and the 0-indexed documents pointed to by the elements of the vector.
`K`	A scalar integer indicating the number of latent topics for the model.
`vocab`	A character vector specifying the vocabulary words associated with the word indices used in *documents*.
`num.iterations`	The number of sweeps of Gibbs sampling over the entire corpus to make.
`num.e.iterations`	For `rtm.em`, the number of iterations in each Gibbs sampling E-step.
`num.m.iterations`	For `rtm.em`, the number of M-step iterations.
`alpha`	The scalar value of the Dirichlet hyperparameter for topic proportions.
`eta`	The scalar value of the Dirichlet hyperparamater for topic multinomials.
`beta`	A length `K` numeric of regression coefficients expressing the relationship between each topic and the probability of link.
`lambda`	For `rtm.em`, the regularization parameter used when estimating beta. *lambda* expresses the number of non-links to simulate among all possible connections between documents.
`initial.beta`	For `rtm.em`, an initial value of `beta` at which to start the EM process.
`trace`	When `trace` is greater than zero, diagnostic messages will be output. Larger values of `trace` imply more messages.
`test.start`	Internal use only.

Details

The Relational Topic Model uses LDA to model the content of documents but adds connections between documents as dependent on the similarity of the distribution of latent topic assignments. (See reference for details).

Only the exponential link probability function is implemented here. Note that the collapsed Gibbs sampler is different than the variational inference procedure proposed in the paper and is extremely experimental.

`rtm.em` provides an EM-wrapper around `rtm.collapsed.gibbs.sampler` which iteratively estimates the regression parameters `beta`.

Value

A fitted model as a list with the same components as returned by `lda.collapsed.gibbs.sampler`.

Author(s)

Jonathan Chang (<`jonchang@fb.com`>)

References

Chang, Jonathan and Blei, David M. Relational Topic Models for Document Networks. Artificial intelligence and statistics. 2009.

See Also

See `lda.collapsed.gibbs.sampler` for a description of the input formats and similar models.

`nubbi.collapsed.gibbs.sampler` is a different kind of model for document networks.

`predictive.link.probability` makes predictions based on the output of this model.

Examples

```
## See demo.

## Not run: demo(rtm)
```

`sampson` *Sampson monk data*

Description

Various relationships between several monks at a monastery collected over time.

Usage

```
data(sampson)
```

Format

`sampson` is a list whose entries are 18x18 matrices representing the pairwise relationships between 18 monks. The names of the monks are given as the row/column names of each matrix.

Each matrix encodes a different relationship (there are a total of 10) described by the corresponding name field of the list.

Source

F. S. Sampson. A novitiate in a period of change: An experimental and case study of social relationships. PhD thesis, Cornell University. 1968.

See Also

`mmsb.collapsed.gibbs.sampler` is an example of a function which can model the structure of this data set.

Examples

```
data(sampson)
```

`slda.predict` *Predict the response variable of documents using an sLDA model.*

Description

This function takes a fitted sLDA model and predicts the value of the response variable for each given document.

Usage

```
slda.predict(documents, topics, model, alpha, eta,
num.iterations = 100, average.iterations = 50, trace = 0L)
```

Arguments

`documents`	A list of document matrices comprising a corpus, in the format described in `lda.collapsed.gibbs.sampler`.
`topics`	A $K \times V$ matrix where each entry is an integer that is the number of times the word (column) has been allocated to the topic (row) (a normalised version of this is sometimes denoted $\beta_{w,k}$ in the literature, see details). The column names should correspond to the words in the vocabulary. The *topics* field from the output of `slda.em` can be used.
`model`	A fitted model relating a document's topic distribution to the response variable. The *model* field from the output of `slda.em` can be used.
`alpha`	The scalar value of the Dirichlet hyperparameter for topic proportions. See references for details.
`eta`	The scalar value of the Dirichlet hyperparamater for topic multinomials.
`num.iterations`	Number of iterations of inference to perform on the documents.
`average.iterations`	Number of samples to average over to produce the predictions.
`trace`	When `trace` is greater than zero, diagnostic messages will be output. Larger values of `trace` imply more messages.

Details

Inference is first performed on the documents by using Gibbs sampling and holding the word-topic matrix $\beta_{w,k}$ constant. Typically for a well-fit model only a small number of iterations are required to obtain good fits for new documents. These topic vectors are then piped through `model` to yield numeric predictions associated with each document.

Value

A numeric vector of the same length as `documents` giving the predictions.

Author(s)

Jonathan Chang (<`jonchang@fb.com`>)

References

Blei, David M. and McAuliffe, John. Supervised topic models. Advances in Neural Information Processing Systems, 2008.

See Also

See `lda.collapsed.gibbs.sampler` for a description of the format of the input data, as well as more details on the model.

See `predictive.distribution` if you want to make predictions about the contents of the documents instead of the response variables.

Examples

```
## The sLDA demo shows an example usage of this function.
## Not run: demo(slda)
```

`top.topic.words` *Get the Top Words and Documents in Each Topic*

Description

This function takes a model fitted using `lda.collapsed.gibbs.sampler` and returns a matrix of the top words in each topic.

Usage

```
top.topic.words(topics, num.words = 20, by.score = FALSE)
top.topic.documents(document_sums, num.documents = 20, alpha = 0.1)
```

Arguments

`topics`	For `top.topic.words`, a $K \times V$ matrix where each entry is a numeric proportional to the probability of seeing the word (column) conditioned on topic (row) (this entry is sometimes denoted $\beta_{w,k}$ in the literature, see details). The column names should correspond to the words in the vocabulary. The *topics* field from the output of `lda.collapsed.gibbs.sampler` can be used.
`num.words`	For `top.topic.words`, the number of top words to return for each topic.
`document_sums`	For `top.topic.documents`, a $K \times D$ matrix where each entry is a numeric proportional to the probability of seeing a topic (row) conditioned on the document (column) (this entry is sometimes denoted $\theta_{d,k}$ in the literature, see details). The *document_sums* field from the output of `lda.collapsed.gibbs.sampler` can be used.
`num.documents`	For `top.topic.documents`, the number of top documents to return for each topic.
`by.score`	If *by.score* is set to `FALSE` (default), then words in each topic will be ranked according to probability mass for each word $\beta_{w,k}$. If *by.score* is `TRUE`, then words will be ranked according to a score defined by $\beta_{w,k}(\log \beta_{w,k} - 1/K \sum_{k'} \log \beta_{w,k'})$.
`alpha`	

Value

For `top.topic.words`, a $num.words \times K$ character matrix where each column contains the top words for that topic.

For `top.topic.documents`, a $num.documents \times K$ integer matrix where each column contains the top documents for that topic. The entries in the matrix are column-indexed references into `document_sums`.

Author(s)

Jonathan Chang (`<jonchang@fb.com>`)

References

Blei, David M. and Ng, Andrew and Jordan, Michael. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

See Also

`lda.collapsed.gibbs.sampler` for the format of *topics*.

`predictive.distribution` demonstrates another use for a fitted topic matrix.

Examples

```
## From demo(lda).

data(cora.documents)
data(cora.vocab)

K <- 10 ## Num clusters
result <- lda.collapsed.gibbs.sampler(cora.documents,
                                      K,  ## Num clusters
                                      cora.vocab,
                                      25,  ## Num iterations
                                      0.1,
                                      0.1)

## Get the top words in the cluster
top.words <- top.topic.words(result$topics, 5, by.score=TRUE)

## top.words:
##      [,1]             [,2]        [,3]       [,4]            [,5]
## [1,] "decision"       "network"   "planning" "learning"      "design"
## [2,] "learning"       "time"      "visual"   "networks"      "logic"
## [3,] "tree"           "networks"  "model"    "neural"        "search"
## [4,] "trees"          "algorithm" "memory"   "system"        "learning"
## [5,] "classification" "data"      "system"   "reinforcement" "systems"
##      [,6]         [,7]       [,8]           [,9]           [,10]
## [1,] "learning"   "models"   "belief"       "genetic"      "research"
## [2,] "search"     "networks" "model"        "search"       "reasoning"
## [3,] "crossover"  "bayesian" "theory"       "optimization" "grant"
## [4,] "algorithm"  "data"     "distribution" "evolutionary" "science"
## [5,] "complexity" "hidden"   "markov"       "function"     "supported"
```

`word.counts` *Compute Summary Statistics of a Corpus*

Description

These functions compute summary statistics of a corpus. `word.counts` computes the word counts for a set of documents, while `documents.length` computes the length of the documents in a corpus.

Usage

```
word.counts(docs, vocab = NULL)

document.lengths(docs)
```

Arguments

`docs`	A list of matrices specifying the corpus. See `lda.collapsed.gibbs.sampler` for details on the format of this variable.
`vocab`	An optional character vector specifying the levels (i.e., labels) of the vocabulary words. If unspecified (or `NULL`), the levels will be automatically inferred from the corpus.

Value

`word.counts` returns an object of class '`table`' which contains counts for the number of times each word appears in the input corpus. If *vocab* is specified, then the levels of the table will be set to *vocab*. Otherwise, the levels are automatically inferred from the corpus (typically integers *0:(V-1)*, where *V* indicates the number of unique words in the corpus).

`documents.length` returns a integer vector of length `length(docs)`, each entry of which corresponds to the *length* (sum of the counts of all features) of each document in the corpus.

Author(s)

Jonathan Chang (`<jonchang@fb.com>`)

See Also

`lda.collapsed.gibbs.sampler` for the input format of these functions.

`read.documents` and `lexicalize` for ways of generating the input to these functions.

`merge.documents` for operations on a corpus.

Examples

```
## Load the cora dataset.
data(cora.vocab)
data(cora.documents)

## Compute word counts using raw feature indices.
wc <- word.counts(cora.documents)
head(wc)
##   0   1   2   3   4   5
## 136 876  14 111  19  29

## Recompute them using the levels defined by the vocab file.
wc <- word.counts(cora.documents, cora.vocab)
head(wc)
##    computer  algorithms discovering    patterns      groups     protein
##         136         876          14         111          19          29

head(document.lengths(cora.documents))
## [1] 64 39 76 84 52 24
```