Package 'mallet'

February 20, 2015

Type Package

Title A wrapper around the Java machine learning tool MALLET

Version 1.0

Date 2013-07-18

Author David Mimno

Maintainer David Mimno <mimno@cornell.edu>

Description This package allows you to train topic models in mallet and load results directly into R.

License MIT + file LICENSE

SystemRequirements java

Depends rJava

NeedsCompilation no

Repository CRAN

Date/Publication 2013-08-09 16:46:59

R topics documented:

mallet-package 2
mallet.doc.topics 3
mallet.import 3
mallet.read.dir 4
mallet.subset.topic.words 5
mallet.top.words 6
mallet.topic.hclust 6
mallet.topic.labels 7
mallet.topic.words 7
mallet.word.freqs 8
MalletLDA 8

Index **11**

mallet-package *An R wrapper for the Mallet topic modeling package*

Description

This package provides an interface to the Java implementation of latent Dirichlet allocation in the Mallet machine learning package. Mallet has many functions, this wrapper focuses on the topic modeling sub-package written by David Mimno. The package uses the `rJava` package to connect to a JVM.

Details

Package:	mallet
Type:	Package
Version:	1.0
Date:	2013-08-08
License:	MIT

Create a topic model trainer: `MalletLDA`

Load documents from disk and import them: `mallet.read.dir` `mallet.import`

Get info about word frequencies: `mallet.word.freqs`

Get trained model parameters: `mallet.doc.topics` `mallet.topic.words` `mallet.subset.topic.words`

Reports on topic words: `mallet.top.words` `mallet.topic.labels`

Clustering of topics: `mallet.topic.hclust`

Author(s)

Maintainer: David Mimno

References

The model, Latent Dirichlet allocation (LDA): *David M Blei, Andrew Ng, Michael Jordan. Latent Dirichlet Allocation. J. of Machine Learning Research, 2003.*

The Java toolkit: *Andrew Kachites McCallum. The Mallet Toolkit. 2002.*

Details of the fast sparse Gibbs sampling algorithm: *Limin Yao, David Mimno, Andrew McCallum. Streaming Inference for Latent Dirichlet Allocation. KDD, 2009.*

Hyperparameter optimization: *Hanna Wallach, David Mimno, Andrew McCallum. Rethinking LDA: Why Priors Matter. NIPS, 2010.*

mallet.doc.topics *Retrieve a matrix of topic weights for every document*

Description

This function returns a matrix with one row for every document and one column for every topic.

Usage

```
mallet.doc.topics(topic.model, normalized, smoothed)
```

Arguments

topic.model	The model returned by MalletLDA
normalized	If true, normalize the rows so that each document sums to one. If false, values will be integers (possibly plus the smoothing constant) representing the actual number of words of each topic in the documents.
smoothed	If true, add the smoothing parameter for the model (initial value specified as alpha.sum in MalletLDA). If false, many values will be zero.

mallet.import *Import text documents into Mallet format*

Description

This function takes an array of document IDs and text files (as character strings) and converts them into a Mallet instance list.

Usage

```
mallet.import(id.array, text.array, stoplist.file, preserve.case, token.regexp)
```

Arguments

id.array	An array of document IDs.
text.array	An array of text strings to use as documents. The type of the array must be character.
stoplist.file	The name of a file containing stopwords (words to ignore), one per line. If the file is not in the current working directory, you may need to include a full path.
preserve.case	By default, the input text is converted to all lowercase.
token.regexp	A quoted string representing a regular expression that defines a token. The default is one or more unicode letter: "[\\p{L}]+". Note that special characters must have double backslashes.

See Also

mallet.word.freqs returns term and document frequencies, which may be useful in selecting stopwords.

Examples

```
## Not run:
mallet.instances <- mallet.import(documents$id, documents$text, "en.txt",
     token.regexp = "\\p{L}[\\p{L}\\p{P}]+\\p{L}")

## End(Not run)
```

---

mallet.read.dir *Import documents from a directory into Mallet format*

Description

This function takes a directory path as its only argument and returns a data.frame() with two columns: <id> & <text>, which can be passed to the mallet.import function. This data.frame() has as many rows as there are files in the Dir.

Usage

```
mallet.read.dir(Dir)
```

Arguments

Dir	The path to a directory containing one document per file.

Note

This function was contributed to RMallet by Dan Bowen.

See Also

mallet.import

Examples

```
## Not run:
documents <- mallet.read.dir(Dir)
mallet.instances <- mallet.import(documents$id, documents$text, "en.txt",
     token.regexp = "\\p{L}[\\p{L}\\p{P}]+\\p{L}")

## End(Not run)
```

`mallet.subset.topic.words` *Estimate topic-word distributions from a sub-corpus*

Description

This function returns a matrix of word probabilities for each topic similar to `mallet.topic.words`, but estimated from a subset of the documents in the corpus. The model assumes that topics are the same no matter where they are used, but we know this is often not the case. This function lets us test whether some words are used more or less than we expect in a particular set of documents.

Usage

```
mallet.subset.topic.words(topic.model, subset.docs, normalized=FALSE, smoothed=FALSE)
```

Arguments

`topic.model`	The model returned by `MalletLDA`
`subset.docs`	An array of TRUE/FALSE values specifying which documents should be used and which should be ignored.
`normalized`	If true, normalize the rows so that each topic sums to one. If false, values will be integers (possibly plus the smoothing constant) representing the actual number of words of each type in the topics.
`smoothed`	If true, add the smoothing parameter for the model (initial value specified as `beta` in `MalletLDA`). If false, many values will be zero.

See Also

`mallet.topic.words`

Examples

```
## Not run:
nips.topic.words <- mallet.subset.topic.words(topic.model, documents$class == "NIPS",
     smoothed=T, normalized=T)

## End(Not run)
```

mallet.top.words *Get the most probable words and their probabilities for one topic.*

Description

This function returns a data frame with two columns, one containing the most probable words as character values, the second containing the weight assigned to that word in the word weights vector you supplied.

Usage

```
mallet.top.words(topic.model, word.weights, num.top.words)
```

Arguments

topic.model	The model returned by MalletLDA
word.weights	A vector of word weights for one topic, usually a row from the topic.words matrix from mallet.topic.words.
num.top.words	The number of most probable words to return. If not specified, defaults to 10.

mallet.topic.hclust *Return a hierarchical clustering of topics*

Description

Returns a hierarchical clustering of topics that can be plotted as a dendrogram. There are two ways of measuring topic similarity: topics may contain the some of the same words, or the may appear in some of the same documents. The balance parameter allows you to interpolate between the similarities determined by these two methods.

Usage

```
mallet.topic.hclust(doc.topics, topic.words, balance)
```

Arguments

doc.topics	A documents by topics matrix of topic probabilities.
topic.words	A topics by words matrix of word probabilities.
balance	A value between 0.0 (use only document-level similarity) and 1.0 (use only word-level similarity).

See Also

This function uses data matrices from mallet.doc.topics and mallet.topic.words

Examples

```
## Not run:
topic.labels <- mallet.topic.labels(topic.model, topic.words, 3)
plot(mallet.topic.hclust(doc.topics, topic.words, 0.3), labels=topic.labels)

## End(Not run)
```

`mallet.topic.labels` *Get strings containing the most probable words for each topic*

Description

This function returns a vector of strings, one for each topic, with the most probable words in that topic separated by spaces.

Usage

```
mallet.topic.labels(topic.model, topic.words, num.top.words)
```

Arguments

`topic.model`	The model returned by `MalletLDA`
`topic.words`	The matrix of topic-word weights returned by `mallet.topic.words`
`num.top.words`	The number of words to include for each topic

See Also

`mallet.topic.words` produces topic-word weights. `mallet.top.words` produces a data frame for a single topic.

`mallet.topic.words` *Retrieve a matrix of words weights for topics*

Description

This function returns a matrix with one row for every topic and one column for every word in the vocabulary.

Usage

```
mallet.topic.words(topic.model, normalized, smoothed)
```

Arguments

topic.model	The model returned by MalletLDA
normalized	If true, normalize the rows so that each topic sums to one. If false, values will be integers (possibly plus the smoothing constant) representing the actual number of words of each type in the topics.
smoothed	If true, add the smoothing parameter for the model (initial value specified as beta in MalletLDA). If false, many values will be zero.

---

mallet.word.freqs *Descriptive statistics of word frequencies*

Description

This method returns a data frame with one row for each unique vocabulary word, and three columns: the word as a character value, the total number of tokens of that word type, and the total number of documents that contain that word at least once. This information can be useful in identifying candidate stopwords.

Usage

```
mallet.word.freqs(topic.model)
```

Arguments

topic.model	A Mallet topic trainer returned by MalletLDA

See Also

MalletLDA

MalletLDA *Create a Mallet topic model trainer*

Description

This function creates a java cc.mallet.topics.RTopicModel object that wraps a Mallet topic model trainer java object, cc.mallet.topics.ParallelTopicModel. Note that you can call any of the methods of this java object as properties. In the example below, I make a call directly to the topic.model$setAlphaOptimization(20 java method, which passes this update to the model itself.

Usage

```
MalletLDA(num.topics, alpha.sum, beta)
```

Arguments

num.topics	The number of topics to use. If not specified, this defaults to 10.
alpha.sum	This is the magnitude of the Dirichlet prior over the topic distribution of a document. The default value is 5.0. With 10 topics, this setting leads to a Dirichlet with parameter $\alpha_k = 0.5$. You can intuitively think of this parameter as a number of "pseudo-words", divided evenly between all topics, that are present in every document no matter how the other words are allocated to topics. This is an initial value, which may be changed during training if hyperparameter optimization is active.
beta	This is the per-word weight of the Dirichlet prior over topic-word distributions. The magnitude of the distribution (the sum over all words of this parameter) is determined by the number of words in the vocabulary. Again, this value may change due to hyperparameter optimization.

Examples

```
## Not run:
library(mallet)

## Create a wrapper for the data with three elements, one for each column.
##  R does some type inference, and will guess wrong, so give it hints with "colClasses".
##  Note that "id" and "text" are special fields -- mallet will look there for input.
##  "class" is arbitrary. We will only use that field on the R side.
documents <- read.table("nips_cvpr.txt", col.names=c("id", "class", "text"),
      colClasses=rep("character", 3), sep="\t", quote="")

## Create a mallet instance list object. Right now I have to specify the stoplist
##  as a file, I can't pass in a list from R.
## This function has a few hidden options (whether to lowercase, how we
##   define a token). See ?mallet.import for details.
mallet.instances <- mallet.import(documents$id, documents$text, "en.txt",
     token.regexp = "\\p{L}[\\p{L}\\p{P}]+\\p{L}")

## Create a topic trainer object.
topic.model <- MalletLDA(num.topics=20)

## Load our documents. We could also pass in the filename of a
##  saved instance list file that we build from the command-line tools.
topic.model$loadDocuments(mallet.instances)

## Get the vocabulary, and some statistics about word frequencies.
##  These may be useful in further curating the stopword list.
vocabulary <- topic.model$getVocabulary()
word.freqs <- mallet.word.freqs(topic.model)

## Optimize hyperparameters every 20 iterations,
##  after 50 burn-in iterations.
topic.model$setAlphaOptimization(20, 50)

## Now train a model. Note that hyperparameter optimization is on, by default.
##  We can specify the number of iterations. Here we'll use a large-ish round number.
```

```
topic.model$train(200)

## NEW: run through a few iterations where we pick the best topic for each token,
##  rather than sampling from the posterior distribution.
topic.model$maximize(10)

## Get the probability of topics in documents and the probability of words in topics.
## By default, these functions return raw word counts. Here we want probabilities,
##  so we normalize, and add "smoothing" so that nothing has exactly 0 probability.
doc.topics <- mallet.doc.topics(topic.model, smoothed=T, normalized=T)
topic.words <- mallet.topic.words(topic.model, smoothed=T, normalized=T)

## What are the top words in topic 7?
##  Notice that R indexes from 1, so this will be the topic that mallet called topic 6.
mallet.top.words(topic.model, topic.words[7,])

## Show the first few documents with at least 5
head(documents[ doc.topics[7,] > 0.05 & doc.topics[10,] > 0.05, ])

## How do topics differ across different sub-corpora?
nips.topic.words <- mallet.subset.topic.words(topic.model, documents$class == "NIPS",
        smoothed=T, normalized=T)
cvpr.topic.words <- mallet.subset.topic.words(topic.model, documents$class == "CVPR",
        smoothed=T, normalized=T)

## How do they compare?
mallet.top.words(topic.model, nips.topic.words[10,])
mallet.top.words(topic.model, cvpr.topic.words[10,])

## End(Not run)
```

Index

mallet *(*mallet-package*)*, 2
mallet-package, 2
mallet.doc.topics, *2*, 3, *6*
mallet.import, *2*, 3, *4*
mallet.read.dir, *2*, 4
mallet.subset.topic.words, *2*, 5
mallet.top.words, *2*, 6, *7*
mallet.topic.hclust, *2*, 6
mallet.topic.labels, *2*, 7
mallet.topic.words, *2*, *5–7*, 7
mallet.word.freqs, *2*, *4*, 8
MalletLDA, *2*, *8*, 8