Package: rollinglda (via r-universe)

September 13, 2024

Type Package

Title Construct Consistent Time Series from Textual Data

Version 0.1.3

Date 2023-11-27

Description A rolling version of the Latent Dirichlet Allocation, see Rieger et al. (2021) <doi:10.18653/v1/2021.findings-emnlp.201>. By a sequential approach, it enables the construction of LDA-based time series of topics that are consistent with previous states of LDA models. After an initial modeling, updates can be computed efficiently, allowing for real-time monitoring and detection of events or structural breaks.

URL https://github.com/JonasRieger/rollinglda

BugReports https://github.com/JonasRieger/rollinglda/issues

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.0.0), ldaPrototype (>= 0.3.0)

Imports checkmate (>= 1.8.5), data.table (>= 1.11.2), lubridate, stats, tosca (>= 0.2-0), utils

Suggests covr, testthat

RoxygenNote 7.2.0

LazyData true

Repository https://jonasrieger.r-universe.dev

RemoteUrl https://github.com/jonasrieger/rollinglda

RemoteRef HEAD

RemoteSha b249e2a8efe14b223ea37ae2017e48d63415eb38

Contents

`rollinglda-package` *rollinglda: Construct Consistent Time Series from Textual Data*

Description

RollingLDA is a rolling version of the Latent Dirichlet Allocation (LDA). By a sequential approach, it enables the construction of LDA-based time series of topics that are consistent with previous states of LDA models. After an initial modeling, updates can be computed efficiently, allowing for real-time monitoring and detection of events or structural breaks.
For bug reports and feature requests please use the issue tracker: https://github.com/JonasRieger/rollinglda/issues. Also have a look at the (detailed) example at https://github.com/JonasRieger/rollinglda.

Data

`economy` Example Dataset (576 articles from Wikinews) for testing.

Constructor

`as.RollingLDA` RollingLDA objects used in this package.

Getter

`getChunks` Getter for `RollingLDA` objects.

Modeling

`RollingLDA` Performing the method from scratch.
`updateRollingLDA` Performing updates on `RollingLDA` objects.

Author(s)

Maintainer: Jonas Rieger <jonas.rieger@tu-dortmund.de> (ORCID)

References

Rieger, Jonas, Carsten Jentsch and Jörg Rahnenführer (2021). "RollingLDA: An Update Algorithm of Latent Dirichlet Allocation to Construct Consistent Time Series from Textual Data". *EMNLP Findings 2021*. URL doi:10.18653/v1/2021.findingsemnlp.201.

See Also

Useful links:

- https://github.com/JonasRieger/rollinglda
- Report bugs at https://github.com/JonasRieger/rollinglda/issues

---

`as.RollingLDA` *RollingLDA Object*

Description

Constructor for RollingLDA objects used in this package. The function may be useful to create a RollingLDA object out of a standard `LDA` object to use it as initial model and update it using `updateRollingLDA`.

Usage

```
as.RollingLDA(x, id, lda, docs, dates, vocab, chunks, param)

is.RollingLDA(obj, verbose = FALSE)
```

Arguments

`x`	[named `list`] `RollingLDA` object. Alternatively each element can be passed for individual results. Individually set elements overwrite elements from `x`.
`id`	[`character(1)`] Name for the computation/model.
`lda`	[named `list`] `LDA` object.
`docs`	[named `list`] Texts in a preprocessed format. See `LDAprep`.
`dates`	[`(un)named Date`] Dates of the texts. If unnamed, it must match the order of docs.
`vocab`	[`character`] Vocabularies.
`chunks`	[`data.table`] with specifications for each model chunk `chunk.id` [`integer`] Index counting up starting with `0`. `start.date` [`Date`] Minimum of each chunk's dates. `end.date` [`Date`] Maximum of each chunk's dates. `memory` [`Date`] Date from which texts are considered as memory. `n` [`integer`] Number of fitted texts. `n.dicsarded` [`integer`] Number of lost texts through preprocessing.

n.memory [integer] Number of texts considered as memory.

n.vocab [integer] Number of vocabularies (monotonously increasing).

If not passed, lda is interpreted as initialization chunk.

param [named list(4)]
Parameters of the object, i.e. parameters for future updates fitted on the to be created model. List always should contain names "vocab.abs", "vocab.rel", "vocab.fallback" and "doc.abs".

obj [R object]
Object to test.

verbose [logical(1)]
Should test information be given in the console?

Details

If you call as.RollingLDA on an object x which already is of the structure of an RollingLDA object (in particular a RollingLDA object itself), the additional arguments id, param, ... may be used to override the specific elements.

Value

[named list] RollingLDA object.

See Also

Other RollingLDA functions: RollingLDA(), getChunks(), updateRollingLDA()

Examples

```
roll_lda = RollingLDA(texts = economy_texts,
                      dates = economy_dates,
                      chunks = "quarter",
                      memory = "3 quarter",
                      init = "2008-07-03",
                      K = 10,
                      type = "lda")

is.RollingLDA(roll_lda, verbose = TRUE)
getID(roll_lda)
roll_lda = as.RollingLDA(roll_lda, id = "newID")
getID(roll_lda)
```

economy *A Snippet of the Economy Dataset from toscaData*

Description

Example Dataset from Wikinews consisting of 576 articles. It can be used to familiarize with the functions offered by this package.

Usage

```
data(economy_texts)

data(economy_dates)
```

Format

`economy_texts` is a named list of tokenized texts of length 576.

`economy_dates` is

An object of class `Date` of length 576.

Source

https://github.com/Docma-TU/toscaData

getChunks *Getter for RollingLDA*

Description

Returns the corresponding element of a `RollingLDA` object.

Usage

```
getChunks(x)

getNames(x)

getDates(x, names, inverse)

getDocs(x, names, inverse)

getVocab(x)

## S3 method for class 'RollingLDA'
getLDA(x, job, reduce, all)
```

```
## S3 method for class 'RollingLDA'
getID(x)

## S3 method for class 'RollingLDA'
getParam(x)
```

Arguments

x	[named list] RollingLDA object.
names	[character] Names of the requested items (dates or docs). Default are all names.
inverse	[logical(1)] Should all items except those with the given names be returned? Default is FALSE.
job	not implemented for RollingLDA object. See getLDA
reduce	not implemented for RollingLDA object. See getLDA
all	not implemented for RollingLDA object. See getLDA

Value

The requested element of a RollingLDA object.

See Also

Other RollingLDA functions: RollingLDA(), as.RollingLDA(), updateRollingLDA()

RollingLDA *RollingLDA*

Description

Performs a rolling version of Latent Dirichlet Allocation.

Usage

```
RollingLDA(...)

## Default S3 method:
RollingLDA(
  texts,
  dates,
  chunks,
  memory,
  vocab.abs = 5L,
```

```
  vocab.rel = 0,
  vocab.fallback = 100L,
  doc.abs = 0L,
  memory.fallback = 0L,
  init,
  type = c("ldaprototype", "lda"),
  id,
  ...
)
```

Arguments

...	additional arguments passed to LDARep or LDAPrototype, respectively. Default parameters are alpha = eta = 1/K and num.iterations = 200. There is no default for K.
texts	[named list] Tokenized texts.
dates	[(un)named Date] Dates of the tokenized texts. If unnamed, it must match the order of texts.
chunks	[Date or character(1)] Sorted dates of the beginnings of each chunk to be modeled after the initial model. If passed as character, dates are determined by passing init plus one day as from argument, max(dates) as to argument and chunks as by argument in seq.Date.
memory	[Date, character(1) or integer(1)] Sorted dates of the beginnings of each chunk's memory. If passed as character, dates are determined by using the dates of the beginnings of each chunk and substracting the given time interval in memory passing it as by argument in seq.Date. If passed as integer/numeric, the dates are determined by going backwards the modeled texts chronologically and taking the date of the text at position memory.
vocab.abs	[integer(1)] An absolute lower bound limit for which words are taken into account. All words are considered in the vocabularies that have a count higher than vocab.abs over all texts and at the same time a higher relative frequency than vocab.rel. Default is 5.
vocab.rel	[0,1] A relative lower bound limit for which words are taken into account. See also vocab.abs. Default is 0.
vocab.fallback	[integer(1)] An absolute lower bound limit for which words are taken into account. All words are considered in the vocabularies that have a count higher than vocab.fallback over all texts even if they might not have a higher relative frequency than vocab.rel. Default is 100.
doc.abs	[integer(1)] An absolute lower bound limit for which texts are taken into account. All texts

are considered for modeling that have more words (subsetted to words occurring in the vocabularies) than doc.abs. Default is 0.

memory.fallback
: [integer(1)]
If there are no texts as memory in a certain chunk, memory is determined by going backwards the modeled texts chronologically and taking the date of the text at position memory.fallback. Default is 0, which means "end the fitting".

init
: [Date(1) or integer(1)]
Date up to which the initial model should be computed. This parameter is needed/used only if chunks is passed as character. Otherwise the initial model is computed up to the first date in chunks minus one day. If init is passed as integer/numeric, the init lowest date from dates is selected.

type
: [character(1)]
One of "ldaPrototype" or "lda" specifying whether a LDAProtoype or standard LDA should be modeled as initial model. Default is "ldaprototype".

id
: [character(1)]
Name for the computation/model.

Details

The function first computes a initial LDA model (using LDARep or LDAPrototype). Afterwards it models temporal chunks of texts with a specified memory for initialization of each model chunk.

The function returns a RollingLDA object. You can receive results and all other elements of this object with getter functions (see getChunks).

Value

[named list] with entries

id [character(1)] See above.

lda LDA object of the fitted RollingLDA.

docs [named list] with modeled texts in a preprocessed format. See LDAprep.

dates [named Date] with dates of the modeled texts.

vocab [character] with the vocabularies considered for modeling.

chunks [data.table] with specifications for each model chunk.

param [named list] with parameter specifications for vocab.abs [integer(1)], vocab.rel [0,1], vocab.fallback [integer(1)] and doc.abs [integer(1)]. See above for explanation.

See Also

Other RollingLDA functions: as.RollingLDA(), getChunks(), updateRollingLDA()

Examples

```
roll_lda = RollingLDA(texts = economy_texts,
                      dates = economy_dates,
                      chunks = "quarter",
                      memory = "3 quarter",
                      init = "2008-07-03",
                      K = 10,
                      type = "lda")

roll_lda
getChunks(roll_lda)
getLDA(roll_lda)


roll_proto = RollingLDA(texts = economy_texts,
                        dates = economy_dates,
                        chunks = "quarter",
                        memory = "3 quarter",
                        init = "2007-07-03",
                        K = 10,
                        n = 12,
                        pm.backend = "socket",
                        ncpus = 2)

roll_proto
getChunks(roll_proto)
getLDA(roll_proto)
```

`updateRollingLDA` *Updating an existing RollingLDA object*

Description

Performs an update of an existing object consisting of a rolling version of Latent Dirichlet Allocation.

Usage

```
updateRollingLDA(
  x,
  texts,
  dates,
  chunks,
  memory,
  param = getParam(x),
  compute.topics = TRUE,
  memory.fallback = 0L,
```

```
  ...
)

## S3 method for class 'RollingLDA'
RollingLDA(
  x,
  texts,
  dates,
  chunks,
  memory,
  param = getParam(x),
  compute.topics = TRUE,
  memory.fallback = 0L,
  ...
)
```

Arguments

x	[named list] RollingLDA object.
texts	[named list] Tokenized texts.
dates	[(un)named Date] Sorted dates of the tokenized texts. If unnamed, it must match the order of texts.
chunks	[Date or character(1)] Sorted dates of the beginnings of each chunk to be modeled as updates. If passed as character, dates are determined by passing the minimum of dates as from argument, max(dates) as to argument and chunks as by argument in seq.Date. If not passed, all texts are interpreted as one chunk.
memory	[Date, character(1) or integer(1)] Dates of the beginnings of each chunk's memory. If passed as character, dates are determined by using the dates of the beginnings of each chunk and substracting the given time interval in memory passing it as by argument in seq.Date. If passed as integer/numeric, the dates are determined by going backwards the modeled texts chronologically and taking the date of the text at position memory.
param	[named list] with entries (Default is getParam(x)) vocab.abs [integer(1)] An absolute lower bound limit for which words are taken into account. All words are considered in the vocabularies that have a count higher than vocab.abs over all texts and at the same time a higher relative frequency than vocab.rel. vocab.rel [0,1] A relative lower bound limit for which words are taken into account. See also vocab.abs. vocab.fallback [integer(1)] An absolute lower bound limit for which words are taken into account. All words are considered in the vocabularies that have a count higher than vocab.fallback over all texts even if they might not have a higher relative frequency than vocab.rel.

doc.abs [integer(1)] An absolute lower bound limit for which texts are taken into account. All texts are considered for modeling that have more words (subsetted to words occurring in the vocabularies) than doc.abs.

compute.topics [logical(1)]
Should the topic matrix of the LDA model be computed? Default is TRUE.

memory.fallback
[integer(1)]
If there are no texts as memory in a certain chunk, memory is determined by going backwards the modeled texts chronologically and taking the date of the text at position memory.fallback. Default is 0, which means "end the fitting".

... not implemented

Details

The function uses an existing RollingLDA object and models new texts with a specified memory as initialization of the new LDA chunk.

The function returns a RollingLDA object. You can receive results and all other elements of this object with getter functions (see getChunks).

Value

[named list] with entries

id [character(1)] See above.

lda LDA object of the fitted RollingLDA.

docs [named list] with modeled texts in a preprocessed format. See LDAprep

dates [named Date] with dates of the modeled texts.

vocab [character] with the vocabularies considered for modeling.

chunks [data.table] with specifications for each model chunk.

param [named list] with parameter specifications for vocab.abs [integer(1)], vocab.rel [0,1], vocab.fallback [integer(1)] and doc.abs [integer(1)]. See above for explanation.

See Also

Other RollingLDA functions: RollingLDA(), as.RollingLDA(), getChunks()

Examples

```
roll_lda = RollingLDA(texts = economy_texts[economy_dates < "2008-05-01"],
                      dates = economy_dates[economy_dates < "2008-05-01"],
                      chunks = "month",
                      memory = "month",
                      init = 100,
                      K = 10,
                      type = "lda")

# updateRollingLDA = RollingLDA, if first argument is a RollingLDA object
```

```
roll_update = RollingLDA(roll_lda,
                         texts = economy_texts[economy_dates >= "2008-05-01"],
                         dates = economy_dates[economy_dates >= "2008-05-01"],
                         chunks = "month",
                         memory = "month")

roll_update
getChunks(roll_update)
```

Index