Package 'BTM'

March 15, 2020

Type Package

Title Biterm Topic Models for Short Text

Version 0.3

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description Biterm Topic Models find topics in collections of short texts.
It is a word co-occurrence based topic model that learns topics by modeling word-word co-occurrences patterns which are called biterms.
This in contrast to traditional topic models like Latent Dirichlet Allocation and Probabilistic Latent Semantic Analysis
which are word-document co-occurrence topic models.
A biterm consists of two words co-occurring in the same short text window.
This context window can for example be a twitter message, a short answer on a survey, a sentence of a text or a document identifier.
The techniques are explained in detail in the paper 'A Biterm Topic Model For Short Text' by Xiaohui Yan, Jiafeng Guo, Yanyan Lan, Xueqi Cheng (2013) <https://github.com/xiaohuiyan/xiaohuiyan.github.io/blob/master/paper/BTM-WWW13.pdf>.

License Apache License 2.0

URL https://github.com/bnosac/BTM

Encoding UTF-8

LazyData true

Imports Rcpp, utils

Suggests udpipe, data.table

LinkingTo Rcpp

RoxygenNote 6.1.1

SystemRequirements C++11

NeedsCompilation yes

Author Jan Wijffels [aut, cre, cph] (R wrapper),
BNOSAC [cph] (R wrapper),
Xiaohui Yan [ctb, cph] (BTM C++ library)

Repository CRAN

Date/Publication 2020-03-15 09:10:03 UTC

R topics documented:

BTM *Construct a Biterm Topic Model on Short Text*

Description

The Biterm Topic Model (BTM) is a word co-occurrence based topic model that learns topics by modeling word-word co-occurrences patterns (e.g., biterms)

- A biterm consists of two words co-occurring in the same context, for example, in the same short text window.
- BTM models the biterm occurrences in a corpus (unlike LDA models which model the word occurrences in a document).
- It's a generative model. In the generation procedure, a biterm is generated by drawing two words independently from a same topic z. In other words, the distribution of a biterm $b = (wi, wj)$ is defined as: $P(b) = \sum_k P(wi|z) * P(wj|z) * P(z)$ where k is the number of topics you want to extract.
- Estimation of the topic model is done with the Gibbs sampling algorithm. Where estimates are provided for $P(w|k) = phi$ and $P(z) = theta$.

Usage

```
BTM(data, k = 5, alpha = 50/k, beta = 0.01, iter = 1000,
  window = 15, background = FALSE, trace = FALSE, biterms)
```

Arguments

data	a tokenised data frame containing one row per token with 2 columns • the first column is a context identifier (e.g. a tweet id, a document id, a sentence id, an identifier of a survey answer, an identifier of a part of a text) • the second column is a column called of type character containing the sequence of words occurring within the context identifier
k	integer with the number of topics to identify
alpha	numeric, indicating the symmetric dirichlet prior probability of a topic P(z). Defaults to 50/k.
beta	numeric, indicating the symmetric dirichlet prior probability of a word given the topic P(w|z). Defaults to 0.1.
iter	integer with the number of iterations of Gibbs sampling

window	integer with the window size for biterm extraction. Defaults to 15.
background	logical if set to TRUE, the first topic is set to a background topic that equals to the empirical word distribution. This can be used to filter out common words. Defaults to FALSE.
trace	logical indicating to print out evolution of the Gibbs sampling iterations. Defaults to FALSE.
biterms	optionally, your own set of biterms to use for modelling. This argument should be a data.frame with column names doc_id, term1, term2 and cooc, indicating how many times each biterm (as indicated by terms term1 and term2) is occurring within a certain doc_id. The field cooc indicates how many times this biterm happens with the doc_id. Note that doc_id's which are not in data are not allowed, as well as terms (in term1 and term2) which are not also in data. See the examples. If provided, the window argument is ignored and the data argument will only be used to calculate the background word frequency distribution.

Value

an object of class BTM which is a list containing

- model: a pointer to the C++ BTM model
- K: the number of topics
- W: the number of tokens in the data
- alpha: the symmetric dirichlet prior probability of a topic P(z)
- beta: the symmetric dirichlet prior probability of a word given the topic P(w|z)
- iter: the number of iterations of Gibbs sampling
- background: indicator if the first topic is set to the background topic that equals the empirical word distribution.
- theta: a vector with the topic probability p(z) which is determinated by the overall proportions of biterms in it
- phi: a matrix of dimension W x K with one row for each token in the data. This matrix contains the probability of the token given the topic P(w|z). the rownames of the matrix indicate the token w

Note

A biterm is defined as a pair of words co-occurring in the same text window. If you have as an example a document with sequence of words `'A B C B'`, and assuming the window size is set to 3, that implies there are two text windows which can generate biterms namely text window `'A B C'` with biterms `'A B'`, `'B C'`, `'A C'` and text window `'B C B'` with biterms `'B C'`, `'C B'`, `'B B'` A biterm is an unorder word pair where `'B C' = 'C B'`. Thus, the document `'A B C B'` will have the following biterm frequencies:

- 'A B': 1
- 'B C': 3

- 'A C': 1
- 'B B': 1

These biterms are used to create the model.

References

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, Xueqi Cheng. A Biterm Topic Model For Short Text. WWW2013, https://github.com/xiaohuiyan/BTM, https://github.com/xiaohuiyan/xiaohuiyan.github.io/blob/master/paper/BTM-WWW13.pdf

See Also

predict.BTM, terms.BTM, logLik.BTM

Examples

```
library(udpipe)
data("brussels_reviews_anno", package = "udpipe")
x <- subset(brussels_reviews_anno, language == "nl")
x <- subset(x, xpos %in% c("NN", "NNP", "NNS"))
x <- x[, c("doc_id", "lemma")]
model  <- BTM(x, k = 5, alpha = 1, beta = 0.01, iter = 10, trace = TRUE)
model
terms(model)
scores <- predict(model, newdata = x)

## Another small run with first topic the background word distribution
set.seed(123456)
model <- BTM(x, k = 5, beta = 0.01, iter = 10, background = TRUE)
model
terms(model)

##
## You can also provide your own set of biterms to cluster upon
## Example: cluster nouns and adjectives in the neighbourhood of one another
##
library(data.table)
library(udpipe)
x <- subset(brussels_reviews_anno, language == "nl")
x <- head(x, 5500) # take a sample to speed things up on CRAN
biterms <- as.data.table(x)
biterms <- biterms[, cooccurrence(x = lemma,
                                  relevant = xpos %in% c("NN", "NNP", "NNS", "JJ"),
                                  skipgram = 2),
                   by = list(doc_id)]
head(biterms)
set.seed(123456)
x <- subset(x, xpos %in% c("NN", "NNP", "NNS", "JJ"))
x <- x[, c("doc_id", "lemma")]
model <- BTM(x, k = 5, beta = 0.01, iter = 10, background = TRUE,
             biterms = biterms, trace = 10)
```

```
model
terms(model)
bitermset <- terms(model, "biterms")
head(bitermset$biterms, 100)

bitermset$n
sum(biterms$cooc)
```

`logLik.BTM` *Get the likelihood of biterms in a BTM model*

Description

Get the likelihood how good biterms are fit by the BTM model

Usage

```
## S3 method for class 'BTM'
logLik(object, data = terms.BTM(object, type =
  "biterms")$biterms, ...)
```

Arguments

`object`	an object of class BTM as returned by `BTM`
`data`	a data.frame with 2 columns term1 and term2 containing biterms. Defaults to the biterms used to construct the model.
`...`	other arguments not used

Value

a list with elements

- likelihood: a vector with the same number of rows as `data` containing the likelihood of the biterms alongside the BTM model. Calculated as `sum(phi[term1,] * phi[term2,] * theta)`.
- `ll` the sum of the log of the biterm likelihoods

See Also

`BTM`, `predict.BTM`, `terms.BTM`

Examples

```
library(udpipe)
data("brussels_reviews_anno", package = "udpipe")
x <- subset(brussels_reviews_anno, language == "nl")
x <- subset(x, xpos %in% c("NN", "NNP", "NNS"))
x <- x[, c("doc_id", "lemma")]

model  <- BTM(x, k = 5, iter = 5, trace = TRUE)
fit <- logLik(model)
fit$ll
```

`predict.BTM` *Predict function for a Biterm Topic Model*

Description

Classify new text alongside the biterm topic model.

To infer the topics in a document, it is assumed that the topic proportions of a document is driven by the expectation of the topic proportions of biterms generated from the document.

Usage

```
## S3 method for class 'BTM'
predict(object, newdata, type = c("sum_b", "sub_w", "mix"),
  ...)
```

Arguments

`object`	an object of class BTM as returned by `BTM`
`newdata`	a tokenised data frame containing one row per token with 2 columns • the first column is a context identifier (e.g. a tweet id, a document id, a sentence id, an identifier of a survey answer, an identifier of a part of a text) • the second column is a column called of type character containing the sequence of words occurring within the context identifier
`type`	character string with the type of prediction. Either one of 'sum_b', 'sub_w' or 'mix'. Default is set to 'sum_b' as indicated in the paper, indicating to sum over the the expectation of the topic proportions of biterms generated from the document. For the other approaches, please inspect the paper.
`...`	not used

Value

a matrix containing containing P(z|d) - the probability of the topic given the biterms.
The matrix has one row for each unique doc_id (context identifier) which contains words part of the dictionary of the BTM model and has K columns, one for each topic.

References

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, Xueqi Cheng. A Biterm Topic Model For Short Text. WWW2013, https://github.com/xiaohuiyan/BTM, https://github.com/xiaohuiyan/xiaohuiyan.github.io/blob/master/paper/BTM-WWW13.pdf

See Also

BTM, terms.BTM, logLik.BTM

Examples

```
library(udpipe)
data("brussels_reviews_anno", package = "udpipe")
x <- subset(brussels_reviews_anno, language == "nl")
x <- subset(x, xpos %in% c("NN", "NNP", "NNS"))
x <- x[, c("doc_id", "lemma")]
model  <- BTM(x, k = 5, iter = 5, trace = TRUE)
scores <- predict(model, newdata = x, type = "sum_b")
scores <- predict(model, newdata = x, type = "sub_w")
scores <- predict(model, newdata = x, type = "mix")
head(scores)
```

---

terms.BTM *Get highest token probabilities for each topic or get biterms used in the model*

Description

Get highest token probabilities for each topic or get biterms used in the model

Usage

```
## S3 method for class 'BTM'
terms(x, type = c("tokens", "biterms"), threshold = 0,
  top_n = 5, ...)
```

Arguments

x	an object of class BTM as returned by BTM
type	a character string, either 'tokens' or 'biterms'. Defaults to 'tokens'.
threshold	threshold in 0-1 range. Only the terms which are more likely than the threshold are returned for each topic. Only used in case type = 'tokens'.
top_n	integer indicating to return the top n tokens for each topic only. Only used in case type = 'tokens'.
...	not used

Value

Depending on if type is set to 'tokens' or 'biterms' the following is returned:

- If `type='tokens'`: Get the probability of the token given the topic P(w|z). It returns a list of data.frames (one for each topic) where each data.frame contains columns token and probability ordered from high to low. The list is the same length as the number of topics.
- If `type='biterms'`: a list containing 2 elements:
  - `n` which indicates the number of biterms used to train the model
  - `biterms` which is a data.frame with columns term1, term2 and topic, indicating for all biterms found in the data the topic to which the biterm is assigned to

  Note that a biterm is unordered, in the output of `type='biterms'` term1 is always smaller than or equal to term2.

See Also

`BTM`, `predict.BTM`, `logLik.BTM`

Examples

```
library(udpipe)
data("brussels_reviews_anno", package = "udpipe")
x <- subset(brussels_reviews_anno, language == "nl")
x <- subset(x, xpos %in% c("NN", "NNP", "NNS"))
x <- x[, c("doc_id", "lemma")]
model  <- BTM(x, k = 5, iter = 5, trace = TRUE)
terms(model)
terms(model, top_n = 10)
terms(model, threshold = 0.01, top_n = +Inf)
bi <- terms(model, type = "biterms")
str(bi)
```

Index