

# Package ‘topicmodels.etm’

July 22, 2025

**Type** Package

**Title** Topic Modelling in Embedding Spaces

**Version** 0.1.0

**Maintainer** Jan Wijffels <jwijffels@bnosac.be>

## Description

Find topics in texts which are semantically embedded using techniques like word2vec or Glove. This topic modelling technique models each word with a categorical distribution whose natural parameter is the inner product between a word embedding and an embedding of its assigned topic. The techniques are explained in detail in the paper 'Topic Modeling in Embedding Spaces' by Adji B. Dieng, Francisco J. R. Ruiz, David M. Blei (2019), available at <[doi:10.48550/arXiv.1907.04907](https://doi.org/10.48550/arXiv.1907.04907)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**SystemRequirements** LibTorch (<https://pytorch.org/>)

**Depends** R (>= 2.10)

**Imports** graphics, stats, Matrix, torch (>= 0.5.0)

**Suggests** udpipe (>= 0.8.4), word2vec, uwot, tinytest, textplot (>= 0.2.0), ggrepel, ggalt

**RoxygenNote** 7.1.2

**NeedsCompilation** no

**Author** Jan Wijffels [aut, cre, cph] (R implementation),  
BNOSAC [cph] (R implementation),  
Adji B. Dieng [ctb, cph] (original Python implementation in inst/orig),  
Francisco J. R. Ruiz [ctb, cph] (original Python implementation in inst/orig),  
David M. Blei [ctb, cph] (original Python implementation in inst/orig)

**Repository** CRAN

**Date/Publication** 2021-11-08 08:40:02 UTC

Contents

|                         |           |
|-------------------------|-----------|
| as.matrix.ETM . . . . . | 2         |
| ETM . . . . .           | 3         |
| ng20 . . . . .          | 5         |
| plot.ETM . . . . .      | 6         |
| predict.ETM . . . . .   | 7         |
| summary.ETM . . . . .   | 9         |
| <b>Index</b>            | <b>11</b> |

---

|               |                                          |
|---------------|------------------------------------------|
| as.matrix.ETM | <i>Get matrices out of an ETM object</i> |
|---------------|------------------------------------------|

---

Description

- Convenience function to extract
- embeddings of the topic centers
  - embeddings of the words used in the model
  - words emitted by each topic (beta), which is the softmax-transformed inner product of word embedding and topic embeddings

Usage

```
## S3 method for class 'ETM'  
as.matrix(x, type = c("embedding", "beta"), which = c("topics", "words"), ...)
```

Arguments

|       |                                                                                                                                                                                                              |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x     | an object of class ETM                                                                                                                                                                                       |
| type  | character string with the type of information to extract: either 'beta' (words emitted by each topic) or 'embedding' (embeddings of words or topic centers). Defaults to 'embedding'.                        |
| which | a character string with either 'words' or 'topics' to get either the embeddings of the words used in the model or the embedding of the topic centers. Defaults to 'topics'. Only used if type = 'embedding'. |
| ...   | not used                                                                                                                                                                                                     |

Value

a numeric matrix containing, depending on the value supplied in type either the embeddings of the topic centers, the embeddings of the words or the words emitted by each topic

See Also

[ETM](#)

## Examples

```
library(torch)
library(topicmodels.etm)
path <- system.file(package = "topicmodels.etm", "example", "example_etm.ckpt")
model <- torch_load(path)

topic.centers <- as.matrix(model, type = "embedding", which = "topics")
word.embeddings <- as.matrix(model, type = "embedding", which = "words")
topic.terminology <- as.matrix(model, type = "beta")
```

---

ETM

*Topic Modelling in Semantic Embedding Spaces*


---

## Description

ETM is a generative topic model combining traditional topic models (LDA) with word embeddings (word2vec).

- It models each word with a categorical distribution whose natural parameter is the inner product between a word embedding and an embedding of its assigned topic.
- The model is fitted using an amortized variational inference algorithm on top of libtorch.

## Usage

```
ETM(
  k = 20,
  embeddings,
  dim = 800,
  activation = c("relu", "tanh", "softplus", "rrelu", "leakyrelu", "elu", "selu",
    "glu"),
  dropout = 0.5,
  vocab = rownames(embeddings)
)
```

## Arguments

|            |                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| k          | the number of topics to extract                                                                                                                                   |
| embeddings | either a matrix with pretrained word embeddings or an integer with the dimension of the word embeddings. Defaults to 50 if not provided.                          |
| dim        | dimension of the variational inference hyperparameter theta (passed on to <a href="#">nn_linear</a> ). Defaults to 800.                                           |
| activation | character string with the activation function of theta. Either one of 'relu', 'tanh', 'softplus', 'rrelu', 'leakyrelu', 'elu', 'selu', 'glu'. Defaults to 'relu'. |
| dropout    | dropout percentage on the variational distribution for theta (passed on to <a href="#">nn_dropout</a> ). Defaults to 0.5.                                         |

**vocab** a character vector with the words from the vocabulary. Defaults to the rownames of the embeddings argument.

### Value

an object of class ETM which is a torch nn\_module containing o.a.

- num\_topics: the number of topics
- vocab: character vector with the terminology used in the model
- vocab\_size: the number of words in vocab
- rho: The word embeddings
- alphas: The topic embeddings

### Methods

**fit**(data, optimizer, epoch, batch\_size, normalize = TRUE, clip = 0, lr\_anneal\_factor = 4, lr\_anneal\_nonmono) Fit the model on a document term matrix by splitting the data in 70/30 training/test set and updating the model weights.

### Arguments

**data** bag of words document term matrix in dgCMatrix format

**optimizer** object of class torch\_Optimizer

**epoch** integer with the number of iterations to train

**batch\_size** integer with the size of the batch

**normalize** logical indicating to normalize the bag of words data

**clip** number between 0 and 1 indicating to do gradient clipping - passed on to [nn\\_utils\\_clip\\_grad\\_norm](#)

**lr\_anneal\_factor** divide the learning rate by this factor when the loss on the test set is monotonic for at least lr\_anneal\_nonmono training iterations

**lr\_anneal\_nonmono** number of iterations after which learning rate annealing is executed if the loss does not decreases

### References

<https://arxiv.org/pdf/1907.04907.pdf>

### Examples

```
library(torch)
library(topicmodels.etm)
library(word2vec)
library(udpipe)
data(brussels_reviews_anno, package = "udpipe")
##
## Toy example with pretrained embeddings
##
```

```

## a. build word2vec model
x      <- subset(brussels_reviews_anno, language %in% "nl")
x      <- paste.data.frame(x, term = "lemma", group = "doc_id")
set.seed(4321)
w2v     <- word2vec(x = x$lemma, dim = 15, iter = 20, type = "cbow", min_count = 5)
embeddings <- as.matrix(w2v)

## b. build document term matrix on nouns + adjectives, align with the embedding terms
dtm <- subset(brussels_reviews_anno, language %in% "nl" & upos %in% c("NOUN", "ADJ"))
dtm <- document_term_frequencies(dtm, document = "doc_id", term = "lemma")
dtm <- document_term_matrix(dtm)
dtm <- dtm_conform(dtm, columns = rownames(embeddings))
dtm <- dtm[dtm_rowsums(dtm) > 0, ]

## create and fit an embedding topic model - 8 topics, theta 100-dimensional
if (torch::torch_is_installed()) {

  set.seed(4321)
  torch_manual_seed(4321)
  model      <- ETM(k = 8, dim = 100, embeddings = embeddings, dropout = 0.5)
  optimizer  <- optim_adam(params = model$parameters, lr = 0.005, weight_decay = 0.0000012)
  overview   <- model$fit(data = dtm, optimizer = optimizer, epoch = 40, batch_size = 1000)
  scores     <- predict(model, dtm, type = "topics")

  lastbatch  <- subset(overview$loss, overview$loss$batch_is_last == TRUE)
  plot(lastbatch$epoch, lastbatch$loss)
  plot(overview$loss_test)

  ## show top words in each topic
  terminology <- predict(model, type = "terms", top_n = 7)
  terminology

  ##
  ## Toy example without pretrained word embeddings
  ##
  set.seed(4321)
  torch_manual_seed(4321)
  model      <- ETM(k = 8, dim = 100, embeddings = 15, dropout = 0.5, vocab = colnames(dtm))
  optimizer  <- optim_adam(params = model$parameters, lr = 0.005, weight_decay = 0.0000012)
  overview   <- model$fit(data = dtm, optimizer = optimizer, epoch = 40, batch_size = 1000)
  terminology <- predict(model, type = "terms", top_n = 7)
  terminology

}

```

**Description**

Data available at <https://github.com/bnosac-dev/ETM/tree/master/data/20ng>

**Examples**

```
data(ng20)
str(ng20$vocab)
str(ng20$bow_tr$tokens)
str(ng20$bow_tr$counts)
```

---

plot.ETM

---

*Plot functionality for an ETM object*


---

**Description**

Convenience function allowing to plot

- the evolution of the loss on the training / test set in order to inspect training convergence
- the ETM model in 2D dimensional space using a umap projection. This plot uses function [textplot\\_embedding\\_2d](#) from the textplot R package and plots the top\_n most emitted words of each topic and the topic centers in 2 dimensions

**Usage**

```
## S3 method for class 'ETM'
plot(
  x,
  type = c("loss", "topics"),
  which,
  top_n = 4,
  title = "ETM topics",
  subtitle = "",
  encircle = FALSE,
  points = FALSE,
  ...
)
```

**Arguments**

|       |                                                                                                                   |
|-------|-------------------------------------------------------------------------------------------------------------------|
| x     | an object of class ETM                                                                                            |
| type  | character string with the type of plot to generate: either 'loss' or 'topics'                                     |
| which | an integer vector of topics to plot, used in case type = 'topics'. Defaults to all topics. See the example below. |
| top_n | passed on to summary.ETM in order to visualise the top_n most relevant words for each topic. Defaults to 4.       |
| title | passed on to textplot_embedding_2d, used in case type = 'topics'                                                  |

|          |                                                                                             |
|----------|---------------------------------------------------------------------------------------------|
| subtitle | passed on to <code>textplot_embedding_2d</code> , used in case <code>type = 'topics'</code> |
| encircle | passed on to <code>textplot_embedding_2d</code> , used in case <code>type = 'topics'</code> |
| points   | passed on to <code>textplot_embedding_2d</code> , used in case <code>type = 'topics'</code> |
| ...      | arguments passed on to <a href="#">summary.ETM</a>                                          |

**Value**

In case `type` is set to `'topics'`, maps the topic centers and most emitted words for each topic to 2D using [summary.ETM](#) and returns a ggplot object by calling [textplot\\_embedding\\_2d](#).  
For type `'loss'`, makes a base graphics plot and returns invisibly nothing.

**See Also**

[ETM](#), [summary.ETM](#), [textplot\\_embedding\\_2d](#)

**Examples**

```
library(torch)
library(topicmodels.etm)
path <- system.file(package = "topicmodels.etm", "example", "example_etm.ckpt")
model <- torch_load(path)
plot(model, type = "loss")

library(torch)
library(topicmodels.etm)
library(textplot)
library(uwot)
library(ggrepel)
library(ggalt)
path <- system.file(package = "topicmodels.etm", "example", "example_etm.ckpt")
model <- torch_load(path)
plt <- plot(model, type = "topics", top_n = 7, which = c(1, 2, 14, 16, 18, 19),
            metric = "cosine", n_neighbors = 15,
            fast_sgd = FALSE, n_threads = 2, verbose = TRUE,
            title = "ETM Topics example")

plt
```

---

predict.ETM

*Predict functionality for an ETM object.*

---

**Description**

Predict to which ETM topic a text belongs or extract which words are emitted for each topic.

**Usage**

```
## S3 method for class 'ETM'
predict(
  object,
  newdata,
  type = c("topics", "terms"),
  batch_size = nrow(newdata),
  normalize = TRUE,
  top_n = 10,
  ...
)
```

**Arguments**

|                         |                                                                                                                                                                                                                                      |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>object</code>     | an object of class <code>ETM</code>                                                                                                                                                                                                  |
| <code>newdata</code>    | bag of words document term matrix in <code>dgCMatrix</code> format. Only used in case <code>type = 'topics'</code> .                                                                                                                 |
| <code>type</code>       | a character string with either <code>'topics'</code> or <code>'terms'</code> indicating to either predict to which topic a document encoded as a set of bag of words belongs to or to extract the most emitted terms for each topic  |
| <code>batch_size</code> | integer with the size of the batch in order to do chunkwise predictions in chunks of <code>batch_size</code> rows. Defaults to the whole dataset provided in <code>newdata</code> . Only used in case <code>type = 'topics'</code> . |
| <code>normalize</code>  | logical indicating to normalize the bag of words data. Defaults to <code>TRUE</code> similar as the default when building the <code>ETM</code> model. Only used in case <code>type = 'topics'</code> .                               |
| <code>top_n</code>      | integer with the number of most relevant words for each topic to extract. Only used in case <code>type = 'terms'</code> .                                                                                                            |
| <code>...</code>        | not used                                                                                                                                                                                                                             |

**Value**

Returns for

- type `'topics'`: a matrix with topic probabilities of dimension `nrow(newdata)` x the number of topics
- type `'terms'`: a list of `data.frame`'s where each `data.frame` has columns `term`, `beta` and `rank` indicating the `top_n` most emitted terms for that topic. List element 1 corresponds to the top terms emitted by topic 1, element 2 to topic 2 ...

**See Also**

[ETM](#)



## Examples

```
library(torch)
library(topicmodels.etm)
path <- system.file(package = "topicmodels.etm", "example", "example_etm.ckpt")
model <- torch_load(path)

# Get most emitted words for each topic
terminology <- predict(model, type = "terms", top_n = 5)
terminology

# Get topics probabilities for each document
path <- system.file(package = "topicmodels.etm", "example", "example_dtm.rds")
dtm <- readRDS(path)
dtm <- head(dtm, n = 5)
scores <- predict(model, newdata = dtm, type = "topics")
scores
```

---

summary.ETM

*Project ETM embeddings using UMAP*


---

## Description

Uses the uwot package to map the word embeddings and the center of the topic embeddings to a 2-dimensional space

## Usage

```
## S3 method for class 'ETM'
summary(object, type = c("umap"), n_components = 2, top_n = 20, ...)
```

## Arguments

|              |                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------|
| object       | object of class ETM                                                                                                           |
| type         | character string with the type of summary to extract. Defaults to 'umap', no other summary information currently implemented. |
| n_components | the dimension of the space to embed into. Passed on to <a href="#">umap</a> . Defaults to 2.                                  |
| top_n        | passed on to <a href="#">predict.ETM</a> to get the top_n most relevant words for each topic in the 2-dimensional space       |
| ...          | further arguments passed onto <a href="#">umap</a>                                                                            |

## Value

a list with elements

- center: a matrix with the embeddings of the topic centers
- words: a matrix with the embeddings of the words

- `embed_2d`: a data.frame which contains a lower dimensional presentation in 2D of the topics and the `top_n` words associated with the topic, containing columns `type`, `term`, `cluster` (the topic number), `rank`, `beta`, `x`, `y`, `weight`; where `type` is either 'words' or 'centers', `x/y` contain the lower dimensional positions in 2D of the word and `weight` is the emitted beta scaled to the highest beta within a topic where the topic center always gets weight 0.8

## See Also

[umap](#), [ETM](#)

## Examples

```
library(torch)
library(topicmodels.etm)
library(uwot)
path      <- system.file(package = "topicmodels.etm", "example", "example_etm.ckpt")
model     <- torch_load(path)
overview  <- summary(model,
                      metric = "cosine", n_neighbors = 15,
                      fast_sgd = FALSE, n_threads = 1, verbose = TRUE)
overview$center
overview$embed_2d
```

# Index

`as.matrix.ETM`, [2](#)

`ETM`, [2](#), [3](#), [7](#), [8](#), [10](#)

`ng20`, [5](#)

`nn_dropout`, [3](#)

`nn_linear`, [3](#)

`nn_utils_clip_grad_norm_`, [4](#)

`plot.ETM`, [6](#)

`predict.ETM`, [7](#), [9](#)

`summary.ETM`, [7](#), [9](#)

`textplot_embedding_2d`, [6](#), [7](#)

`umap`, [9](#), [10](#)