Package ‘ruimtehol’

November 29, 2020

Type Package

Title Learn Text 'Embeddings' with 'Starspace'

Version 0.3

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description Wraps the 'StarSpace' library <https://github.com/facebookresearch/StarSpace> allowing users to calculate word, sentence, article, document, webpage, link and entity 'embeddings'.
By using the 'embeddings', you can perform text based multi-label classification, find similarities between texts and categories, do collaborative-filtering based recommendation as well as content-based recommendation, find out relations between entities, calculate graph 'embeddings' as well as perform semi-supervised learning and multi-task learning on plain text.
The techniques are explained in detail in the paper: 'StarSpace: Embed All The Things!' by Wu et al. (2017), available at <arXiv:1709.03856>.

License MPL-2.0

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

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports Rcpp (>= 0.11.5), utils, graphics, stats

Suggests udpipe, data.table

LinkingTo Rcpp, BH

RoxygenNote 7.1.1

SystemRequirements C++11

NeedsCompilation yes

Author Jan Wijffels [aut, cre, cph] (R wrapper),
BNOSAC [cph] (R wrapper),
Facebook, Inc. [cph] (Starspace (BSD licensed))

Repository CRAN

Date/Publication 2020-11-29 17:40:02 UTC

R topics documented:

`dekamer` *Dataset from 2017 with Questions and Answers in the Belgium Federal Parliament*

Description

Dataset from 2017 with Questions asked by members of the Belgian Federal Parliament and the Answers provided to these questions.
The dataset was extracted from `http://data.dekamer.be` and contains questions asked by persons in the Belgium Federal parliament and answers given by the departments of the Federal Belgian Ministers.
The language of this dataset provided in this R package has been restricted to Dutch.

The dataset contains the following information:

- doc_id: a unique identifier
- depotdat: the date when the question was registered
- aut_party / aut_person / aut_language: who asked the question and which political party is he/she a member of + the language of the person who asked the question
- question: the question itself (always in Dutch)
- question_theme_main: the main theme of the question
- question_theme: a comma-separated list of all themes the question is about
- answer: the answer given by the department of the minister (always in Dutch)
- answer_deptpres, answer_department, answer_subdepartment: to which ministerial department has the question been raised to and answered by

Source

http://data.dekamer.be, data is provided by www.dekamer.be in the public domain (CC0).

Examples

```
data(dekamer)
str(dekamer)
```

---

dekamer_theme_terminology *Dataset containing relevant terminology for each theme of the* dekamer *dataset*

Description

Dataset containing relevant terminology for each theme of the dekamer dataset

The dataset contains the following information:

- theme: a theme, corresponding to the question_theme_main field in the dekamer dataset
- term: a word which describes the theme
- n: a measure of information indicating how relevant the term is (frequency of occurrence)

Examples

```
data(dekamer_theme_terminology)
str(dekamer_theme_terminology)
```

---

embedding_similarity *Cosine and Inner product based similarity*

Description

Cosine and Inner product based similarity

Usage

```
embedding_similarity(x, y, type = c("cosine", "dot"), top_n = +Inf)
```

Arguments

`x` a matrix with embeddings providing embeddings for words/n-grams/documents/labels as indicated in the rownames of the matrix

`y` a matrix with embeddings providing embeddings for words/n-grams/documents/labels as indicated in the rownames of the matrix

`type` either 'cosine' or 'dot'. If 'dot', returns inner-product based similarity, if 'cosine', returns cosine similarity

`top_n` integer indicating to return only the top n most similar terms from `y` for each row of `x`. If `top_n` is supplied, a data.frame will be returned with only the highest similarities between `x` and `y` instead of all pairwise similarities

Value

By default, the function returns a similarity matrix between the rows of `x` and the rows of `y`. The similarity between row i of `x` and row j of `y` is found in cell `[i,j]` of the returned similarity matrix. If `top_n` is provided, the return value is a data.frame with columns term1, term2, similarity and rank indicating the similarity between the provided terms in `x` and `y` ordered from high to low similarity and keeping only the top_n most similar records.

Examples

```
x <- matrix(rnorm(6), nrow = 2, ncol = 3)
rownames(x) <- c("word1", "word2")
y <- matrix(rnorm(15), nrow = 5, ncol = 3)
rownames(y) <- c("term1", "term2", "term3", "term4", "term5")

embedding_similarity(x, y, type = "cosine")
embedding_similarity(x, y, type = "dot")
embedding_similarity(x, y, type = "cosine", top_n = 1)
embedding_similarity(x, y, type = "dot", top_n = 1)
embedding_similarity(x, y, type = "cosine", top_n = 2)
embedding_similarity(x, y, type = "dot", top_n = 2)
embedding_similarity(x, y, type = "cosine", top_n = +Inf)
embedding_similarity(x, y, type = "dot", top_n = +Inf)
```

`embed_articlespace` *Build a Starspace model for learning the mapping between sentences and articles (articlespace)*

Description

Build a Starspace model for learning the mapping between sentences and articles (articlespace)

Usage

```
embed_articlespace(
  x,
  model = "articlespace.bin",
  early_stopping = 0.75,
  useBytes = FALSE,
  ...
)
```

Arguments

`x`	a data.frame with sentences containing the columns doc_id, sentence_id and token The doc_id is just an article or document identifier, the sentence_id column is a character field which contains words which are separated by a space and should not contain any tab characters
`model`	name of the model which will be saved, passed on to `starspace`
`early_stopping`	the percentage of the data that will be used as training data. If set to a value smaller than 1, 1-`early_stopping` percentage of the data which will be used as the validation set and early stopping will be executed. Defaults to 0.75.
`useBytes`	set to TRUE to avoid re-encoding when writing out train and/or test files. See `writeLines` for details
`...`	further arguments passed on to `starspace` except file, trainMode and fileFormat

Value

an object of class `textspace` as returned by `starspace`.

Examples

```
library(udpipe)
data(brussels_reviews_anno, package = "udpipe")
x <- subset(brussels_reviews_anno, language == "nl")
x$token <- x$lemma
x <- x[, c("doc_id", "sentence_id", "token")]
set.seed(123456789)
model <- embed_articlespace(x, early_stopping = 1,
                            dim = 25, epoch = 25, minCount = 2,
                            negSearchLimit = 1, maxNegSamples = 2)
plot(model)
sentences <- c("ook de keuken zijn zeer goed uitgerust .",
               "het appartement zijn met veel smaak inrichten en zeer proper .")
predict(model, sentences, type = "embedding")
starspace_embedding(model, sentences)

## Not run:
library(udpipe)
data(dekamer, package = "ruimtehol")
dekamer <- subset(dekamer, question_theme_main == "DEFENSIEBELEID")
```

```
x <- udpipe(dekamer$question, "dutch", tagger = "none", parser = "none", trace = 100)
x <- x[, c("doc_id", "sentence_id", "sentence", "token")]
set.seed(123456789)
model <- embed_articlespace(x, early_stopping = 0.8, dim = 15, epoch = 5, minCount = 5)
plot(model)

embeddings <- starspace_embedding(model, unique(x$sentence), type = "document")
dim(embeddings)

sentence <- "Wat zijn de cijfers qua doorstroming van 2016?"
embedding_sentence <- starspace_embedding(model, sentence, type = "document")
mostsimilar <- embedding_similarity(embeddings, embedding_sentence)
head(sort(mostsimilar[, 1], decreasing = TRUE), 3)

## clean up for cran
file.remove(list.files(pattern = ".udpipe$"))

## End(Not run)
```

embed_docspace *Build a Starspace model for content-based recommendation*

Description

Build a Starspace model for content-based recommendation (docspace). For example a user clicks on a webpage and this webpage contains a bunch or words.

Usage

```
embed_docspace(
  x,
  model = "docspace.bin",
  early_stopping = 0.75,
  useBytes = FALSE,
  ...
)
```

Arguments

`x` a data.frame with user interest containing the columns user_id, doc_id and text The user_id is an identifier of a user The doc_id is just an article or document identifier the text column is a character field which contains words which are part of the doc_id, words should be separated by a space and should not contain any tab characters

`model` name of the model which will be saved, passed on to `starspace`

`early_stopping` the percentage of the data that will be used as training data. If set to a value smaller than 1, 1-`early_stopping` percentage of the data which will be used as the validation set and early stopping will be executed. Defaults to 0.75.

useBytes	set to TRUE to avoid re-encoding when writing out train and/or test files. See writeLines for details
...	further arguments passed on to starspace except file, trainMode and fileFormat

Value

an object of class textspace as returned by starspace.

Examples

```
library(udpipe)
data(dekamer, package = "ruimtehol")
data(dekamer_theme_terminology, package = "ruimtehol")
## Which person is interested in which theme (aka document)
x <- table(dekamer$aut_person, dekamer$question_theme_main)
x <- as.data.frame(x)
colnames(x) <- c("user_id", "doc_id", "freq")
## Characterise the themes (aka document)
docs <- split(dekamer_theme_terminology, dekamer_theme_terminology$theme)
docs <- lapply(docs, FUN=function(x){
  data.frame(theme = x$theme[1], text = paste(x$term, collapse = " "),
             stringsAsFactors=FALSE)
})
docs <- do.call(rbind, docs)

## Build a model
train <- merge(x, docs, by.x = "doc_id", by.y = "theme")
train <- subset(train, user_id %in% sample(levels(train$user_id), 4))
set.seed(123456789)
model <- embed_docspace(train, dim = 10, early_stopping = 1)
plot(model)
```

---

embed_entityrelationspace

Build a Starspace model for entity relationship completion

Description

Build a Starspace model for entity relationship completion (graphspace).

Usage

```
embed_entityrelationspace(
  x,
  model = "graphspace.bin",
  early_stopping = 0.75,
  useBytes = FALSE,
  ...
)
```

Arguments

`x` a data.frame with columns entity_head, entity_tail and relation indicating the relation between the head and tail entity

`model` name of the model which will be saved, passed on to `starspace`

`early_stopping` the percentage of the data that will be used as training data. If set to a value smaller than 1, 1-`early_stopping` percentage of the data which will be used as the validation set and early stopping will be executed. Defaults to 0.75.

`useBytes` set to TRUE to avoid re-encoding when writing out train and/or test files. See `writeLines` for details

`...` further arguments passed on to `starspace` except file, trainMode and fileFormat

Value

an object of class `textspace` as returned by `starspace`.

Examples

```
## Example on Freebase - download the data
filename <- paste(
  "https://raw.githubusercontent.com/bnosac-dev/GraphEmbeddings/master/",
  "diffbot_data/FB15k/freebase_mtr100_mte100-train.txt",
  sep = "")
tmpfile <- tempfile(pattern = "freebase_mtr100_mte100_", fileext = "txt")
ok <- suppressWarnings(try(
  download.file(url = filename, destfile = tmpfile),
  silent = TRUE))
if(!inherits(ok, "try-error") && ok == 0){
  ## Build the model on the downloaded data
  x <- read.delim(tmpfile, header = FALSE, nrows = 1000,
                  col.names = c("entity_head", "relation", "entity_tail"),
                  stringsAsFactors = FALSE)
  head(x)

  set.seed(123456789)
  model <- embed_entityrelationspace(x, dim = 50)
  plot(model)

  predict(model, "/m/027rn /location/country/form_of_government")

  ## Also add reverse relation
  x_reverse <- x
  colnames(x_reverse) <- c("entity_tail", "relation", "entity_head")
  x_reverse$relation <- sprintf("REVERSE_%s", x_reverse$relation)

  relations <- rbind(x, x_reverse)
  set.seed(123456789)
  model <- embed_entityrelationspace(relations, dim = 50)
  predict(model, "/m/027rn /location/country/form_of_government")
  predict(model, "/m/06cx9 REVERSE_/location/country/form_of_government")
}
```

```
## cleanup for cran
if(file.exists(tmpfile)) file.remove(tmpfile)
```

embed_pagespace *Build a Starspace model for interest-based recommendation*

Description

Build a Starspace model for interest-based recommendation (pagespace). For example a user clicks on a webpage.

Usage

```
embed_pagespace(
  x,
  model = "pagespace.bin",
  early_stopping = 0.75,
  useBytes = FALSE,
  ...
)
```

Arguments

x	a list where each list element contains a character vector of pages which the user was interested in
model	name of the model which will be saved, passed on to starspace
early_stopping	the percentage of the data that will be used as training data. If set to a value smaller than 1, 1-early_stopping percentage of the data which will be used as the validation set and early stopping will be executed. Defaults to 0.75.
useBytes	set to TRUE to avoid re-encoding when writing out train and/or test files. See writeLines for details
...	further arguments passed on to starspace except file, trainMode and fileFormat

Value

an object of class textspace as returned by starspace.

Examples

```
data(dekamer, package = "ruimtehol")
x <- subset(dekamer, !is.na(question_theme))
x <- strsplit(x$question_theme, ",")
x <- lapply(x, FUN=unique)
str(x)
set.seed(123456789)
model <- embed_pagespace(x, dim = 5, epoch = 5, minCount = 10, label = "__THEME__")
```

```
plot(model)
predict(model, "__THEME__MARINE __THEME__DEFENSIEBELEID")

pagevectors <- as.matrix(model)

mostsimilar <- embedding_similarity(pagevectors,
                                    pagevectors["__THEME__MIGRATIEBELEID", ])
head(sort(mostsimilar[, 1], decreasing = TRUE), 3)
mostsimilar <- embedding_similarity(pagevectors,
                                    pagevectors["__THEME__DEFENSIEBELEID", ])
head(sort(mostsimilar[, 1], decreasing = TRUE), 3)
```

`embed_sentencespace` *Build a Starspace model to be used for sentence embedding*

Description

Build a Starspace model to be used for sentence embedding

Usage

```
embed_sentencespace(
  x,
  model = "sentencespace.bin",
  early_stopping = 0.75,
  useBytes = FALSE,
  ...
)
```

Arguments

`x` a data.frame with sentences containg the columns doc_id, sentence_id and token The doc_id is just an article or document identifier, the sentence_id column is a character field which contains words which are separated by a space and should not contain any tab characters

`model` name of the model which will be saved, passed on to `starspace`

`early_stopping` the percentage of the data that will be used as training data. If set to a value smaller than 1, `1-early_stopping` percentage of the data which will be used as the validation set and early stopping will be executed. Defaults to 0.75.

`useBytes` set to TRUE to avoid re-encoding when writing out train and/or test files. See `writeLines` for details

`...` further arguments passed on to `starspace` except file, trainMode and fileFormat

Value

an object of class `textspace` as returned by `starspace`.

Examples

```
library(udpipe)
data(brussels_reviews_anno, package = "udpipe")
x <- subset(brussels_reviews_anno, language == "nl")
x$token <- x$lemma
x <- x[, c("doc_id", "sentence_id", "token")]
set.seed(123456789)
model <- embed_sentencespace(x, dim = 15, epoch = 15,
                             negSearchLimit = 1, maxNegSamples = 2)
plot(model)
sentences <- c("ook de keuken zijn zeer goed uitgerust .",
               "het appartement zijn met veel smaak inrichten en zeer proper .")
predict(model, sentences, type = "embedding")
starspace_embedding(model, sentences)

## Not run:
library(udpipe)
data(dekamer, package = "ruimtehol")
x <- udpipe(dekamer$question, "dutch", tagger = "none", parser = "none", trace = 100)
x <- x[, c("doc_id", "sentence_id", "sentence", "token")]
set.seed(123456789)
model <- embed_sentencespace(x, dim = 15, epoch = 5, minCount = 5)
plot(model)
predict(model, "Wat zijn de cijfers qua doorstroming van 2016?",
        basedoc = unique(x$sentence))

embeddings <- starspace_embedding(model, unique(x$sentence), type = "document")
dim(embeddings)

sentence <- "Wat zijn de cijfers qua doorstroming van 2016?"
embedding_sentence <- starspace_embedding(model, sentence, type = "document")
mostsimilar <- embedding_similarity(embeddings, embedding_sentence)
head(sort(mostsimilar[, 1], decreasing = TRUE), 3)

## clean up for cran
file.remove(list.files(pattern = ".udpipe$"))

## End(Not run)
```

`embed_tagspace` *Build a Starspace model to be used for classification purposes*

Description

Build a Starspace model to be used for classification purposes

Usage

```
embed_tagspace(
  x,
  y,
  model = "tagspace.bin",
  early_stopping = 0.75,
  useBytes = FALSE,
  ...
)
```

Arguments

x	a character vector of text where tokens are separated by spaces
y	a character vector of classes to predict or a list with the same length of x with several classes for each respective element of x
model	name of the model which will be saved, passed on to starspace
early_stopping	the percentage of the data that will be used as training data. If set to a value smaller than 1, 1-early_stopping percentage of the data which will be used as the validation set and early stopping will be executed. Defaults to 0.75.
useBytes	set to TRUE to avoid re-encoding when writing out train and/or test files. See writeLines for details
...	further arguments passed on to starspace except file, trainMode and fileFormat

Value

an object of class textspace as returned by starspace.

Examples

```
data(dekamer, package = "ruimtehol")
dekamer <- subset(dekamer, depotdat < as.Date("2017-02-01"))
dekamer$text <- strsplit(dekamer$question, "\\W")
dekamer$text <- lapply(dekamer$text, FUN = function(x) x[x != ""])
dekamer$text <- sapply(dekamer$text,
                       FUN = function(x) paste(x, collapse = " "))
dekamer$question_theme_main <- gsub(" ", "-", dekamer$question_theme_main)

set.seed(123456789)
model <- embed_tagspace(x = tolower(dekamer$text),
                        y = dekamer$question_theme_main,
                        early_stopping = 0.8,
                        dim = 10, minCount = 5)
plot(model)
predict(model, "de nmbs heeft het treinaanbod uitgebreid", k = 3)
predict(model, "de migranten komen naar europa, in asielcentra ...")
starspace_embedding(model, "de nmbs heeft het treinaanbod uitgebreid")
starspace_embedding(model, "__label__MIGRATIEBELEID", type = "ngram")

dekamer$question_themes <- gsub(" ", "-", dekamer$question_theme)
```

```
dekamer$question_themes <- strsplit(dekamer$question_themes, split = ",")
set.seed(123456789)
model <- embed_tagspace(x = tolower(dekamer$text),
                        y = dekamer$question_themes,
                        early_stopping = 0.8,
                        dim = 50, minCount = 2, epoch = 50)
plot(model)
predict(model, "de nmbs heeft het treinaanbod uitgebreid")
predict(model, "de migranten komen naar europa , in asielcentra ...")
embeddings_labels <- as.matrix(model, type = "labels")
emb <- starspace_embedding(model, "de nmbs heeft het treinaanbod uitgebreid")
embedding_similarity(emb, embeddings_labels, type = "cosine", top_n = 5)
```

`embed_wordspace` *Build a Starspace model which calculates word embeddings*

Description

Build a Starspace model which calculates word embeddings

Usage

```
embed_wordspace(
  x,
  model = "wordspace.bin",
  early_stopping = 0.75,
  useBytes = FALSE,
  ...
)
```

Arguments

`x` a character vector of text where tokens are separated by spaces

`model` name of the model which will be saved, passed on to `starspace`

`early_stopping` the percentage of the data that will be used as training data. If set to a value smaller than 1, 1-`early_stopping` percentage of the data which will be used as the validation set and early stopping will be executed. Defaults to 0.75.

`useBytes` set to TRUE to avoid re-encoding when writing out train and/or test files. See `writeLines` for details

`...` further arguments passed on to `starspace` except file, trainMode and fileFormat

Value

an object of class `textspace` as returned by `starspace`.

Examples

```
library(udpipe)
data(brussels_reviews, package = "udpipe")
x <- subset(brussels_reviews, language == "nl")
x <- strsplit(x$feedback, "\\W")
x <- lapply(x, FUN = function(x) x[x != ""])
x <- sapply(x, FUN = function(x) paste(x, collapse = " "))
x <- tolower(x)

set.seed(123456789)
model <- embed_wordspace(x, early_stopping = 0.9,
                         dim = 15, ws = 7, epoch = 10, minCount = 5, ngrams = 1,
                         maxTrainTime = 2) ## maxTrainTime only set for CRAN
plot(model)
wordvectors <- as.matrix(model)

mostsimilar <- embedding_similarity(wordvectors, wordvectors["weekend", ])
head(sort(mostsimilar[, 1], decreasing = TRUE), 10)
mostsimilar <- embedding_similarity(wordvectors, wordvectors["vriendelijk", ])
head(sort(mostsimilar[, 1], decreasing = TRUE), 10)
mostsimilar <- embedding_similarity(wordvectors, wordvectors["grote", ])
head(sort(mostsimilar[, 1], decreasing = TRUE), 10)
```

`predict.textspace` *Predict using a Starspace model*

Description

The prediction functionality allows you to retrieve the following types of elements from a Starspace model:

- `generic`: get general Starspace predictions in detail
- `labels`: get similarity of your text to all the labels of the Starspace model
- `embedding`: document embeddings of your text (shorthand for `starspace_embedding`)
- `knn`: k-nearest neighbouring (most similar) elements of the model dictionary compared to your input text (shorthand for `starspace_knn`)

Usage

```
## S3 method for class 'textspace'
predict(
  object,
  newdata,
  type = c("generic", "labels", "knn", "embedding"),
  k = 5L,
  sep = " ",
  basedoc,
  ...
)
```

Arguments

`object` an object of class `textspace` as returned by `starspace` or `starspace_load_model`

`newdata` a data frame with columns `doc_id` and `text` or a character vector with text where the names of the character vector represent an identifier of that text

`type` character string: either 'generic', 'labels', 'embedding', 'knn'. Defaults to 'generic'

`k` integer with the number of predictions to make. Defaults to 5. Only used in case `type` is set to `'generic'` or `'knn'`

`sep` character string used to split `newdata` using boost::split. Only used in case `type` is set to `'generic'`

`basedoc` optional, either a character vector of possible elements to predict or the path to a file in labelDoc format, containing basedocs which are set of possible things to predict, if different than the ones from the training data. Only used in case `type` is set to `'generic'`

`...` not used

Value

The following is returned, depending on the argument `type`:

- In case type is set to `'generic'`: a list, one for each row or element in `newdata`. Each list element is a list with elements
  - doc_id: the identifier of the text
  - text: the character string with the text
  - prediction: data.frame with columns label, label_starspace and similarity indicating the predicted label and the similarity of the text to the label
  - terms: a list with elements basedoc_index and basedoc_terms indicating the position in basedoc and the terms which are part of the dictionary which are used to find the similarity
- In case type is set to `'labels'`: a data.frame is returned namely:
  The data.frame `newdata` where several columns are added, one for each label in the Starspace model. These columns contain the similarities of the text to the label. Similarities are computed with `embedding_similarity` indicating embedding similarities of the text compared to the labels using either cosine or dot product as was used during model training.
- In case type is set to `'embedding'`:
  A matrix of document embeddings, one embedding for each text in `newdata` as returned by `starspace_embedding`. The rownames of this matrix are set to the document identifiers of `newdata`.
- In case type is set to `'knn'`: a list of data.frames, one for each row or element in `newdata`
  Each of these data frames contains the columns doc_id, label, similarity and rank indicating the k-nearest neighbouring (most similar) elements of the model dictionary compared to your input text as returned by `starspace_knn`

Examples

```
data(dekamer, package = "ruimtehol")
dekamer$text <- strsplit(dekamer$question, "\\W")
dekamer$text <- lapply(dekamer$text, FUN = function(x) x[x != ""])
dekamer$text <- sapply(dekamer$text,
                       FUN = function(x) paste(x, collapse = " "))

idx <- sample(nrow(dekamer), size = round(nrow(dekamer) * 0.9))
traindata <- dekamer[idx, ]
testdata <- dekamer[-idx, ]
set.seed(123456789)
model <- embed_tagspace(x = traindata$text,
                        y = traindata$question_theme_main,
                        early_stopping = 0.8,
                        dim = 10, minCount = 5)
scores <- predict(model, testdata)
scores <- predict(model, testdata, type = "labels")
str(scores)
emb <- predict(model, testdata[, c("doc_id", "text")], type = "embedding")
knn <- predict(model, testdata[1:5, c("doc_id", "text")], type = "knn", k=3)


## Not run:
library(udpipe)
data(dekamer, package = "ruimtehol")
dekamer <- subset(dekamer, question_theme_main == "DEFENSIEBELEID")
x <- udpipe(dekamer$question, "dutch", tagger = "none", parser = "none", trace = 100)
x <- x[, c("doc_id", "sentence_id", "sentence", "token")]
set.seed(123456789)
model <- embed_sentencespace(x, dim = 15, epoch = 5, minCount = 5)
scores <- predict(model, "Wat zijn de cijfers qua doorstroming van 2016?",
                  basedoc = unique(x$sentence), k = 3)
str(scores)

#' ## clean up for cran
file.remove(list.files(pattern = ".udpipe$"))

## End(Not run)
```

range.textspace *Get the scale of embedding similarities alongside a Starspace model*

Description

Calculates embedding similarities between 2 embedding matrices and gets the range of resulting similarities.

Usage

```
## S3 method for class 'textspace'
range(
  x,
  from = as.matrix(x),
  to = as.matrix(x, type = "labels"),
  probs = seq(0, 1, by = 0.01),
  breaks = "scott",
  ...
)
```

Arguments

`x`	an object of class `textspace` as returned by `starspace` or `starspace_load_model`
`from`	an embedding matrix. Defaults to the embeddings of all the labels and the words from the model.
`to`	an embedding matrix. Defaults to the embeddings of all the labels.
`probs`	numeric vector of probabilities ranging from 0-1. Passed on to `quantile`
`breaks`	passed on to `hist`
`...`	other parameters passed on to `hist`

Value

a list with elements

- range: the range of the embedding similarities between `from` and `to`
- quantile: the quantiles of the embedding similarities between `from` and `to`
- hist: the histogram of the embedding similarities between `from` and `to`

Examples

```
data(dekamer, package = "ruimtehol")
dekamer <- subset(dekamer, depotdat < as.Date("2017-02-01"))
dekamer$text <- strsplit(dekamer$question, "\\W")
dekamer$text <- lapply(dekamer$text, FUN = function(x) setdiff(x, ""))
dekamer$text <- sapply(dekamer$text,
                       FUN = function(x) paste(x, collapse = " "))
dekamer$question_theme_main <- gsub(" ", "-", dekamer$question_theme_main)

set.seed(123456789)
model <- embed_tagspace(x = tolower(dekamer$text),
                        y = dekamer$question_theme_main,
                        early_stopping = 0.8,
                        dim = 10, minCount = 5)
ranges <- range(model)
ranges$range
ranges$quantile
plot(ranges$hist, main = "Histogram of embedding similarities")
```

starspace — *Interface to Starspace for training a Starspace model*

Description

Interface to Starspace for training a Starspace model, providing raw access to the C++ functionality.

Usage

```
starspace(
  model = "textspace.bin",
  file,
  trainMode = 0,
  fileFormat = c("fastText", "labelDoc"),
  label = "__label__",
  dim = 100,
  epoch = 5,
  lr = 0.01,
  loss = c("hinge", "softmax"),
  margin = 0.05,
  similarity = c("cosine", "dot"),
  negSearchLimit = 50,
  adagrad = TRUE,
  ws = 5,
  minCount = 1,
  minCountLabel = 1,
  ngrams = 1,
  thread = 1,
  ...
)
```

Arguments

`model` the full path to where the model file will be saved. Defaults to ’textspace.bin’.

`file` the full path to the file on disk which will be used for training.

`trainMode` integer with the training mode. Possible values are 0, 1, 2, 3, 4 or 5. Defaults to 0. The use cases are

- 0: tagspace (classification tasks) and search tasks
- 1: pagespace & docspace (interest-based or content-based recommendation)
- 2: articlespace (sentences within document)
- 3: sentence embeddings and entity similarity
- 4: multi-relational graphs
- 5: word embeddings

`fileFormat` either one of ’fastText’ or ’labelDoc’. See the documentation of StarSpace `label` labels prefix (character string identifying how a label is prefixed, defaults to ’__label__’)

`dim` the size of the embedding vectors (integer, defaults to 100)

`epoch` number of epochs (integer, defaults to 5)

`lr` learning rate (numeric, defaults to 0.01)

`loss` loss function (either ’hinge’ or ’softmax’)

`margin` margin parameter in case of hinge loss (numeric, defaults to 0.05)

`similarity` cosine or dot product similarity in cas of hinge loss (character, defaults to ’cosine’)

`negSearchLimit` number of negatives sampled (integer, defaults to 50)

`adagrad` whether to use adagrad in training (logical)

`ws` the size of the context window for word level training - only used in trainMode 5 (integer, defaults to 5)

`minCount` minimal number of word occurences for being part of the dictionary (integer, defaults to 1 keeping all words)

`minCountLabel` minimal number of label occurences for being part of the dictionary (integer, defaults to 1 keeping all labels)

`ngrams` max length of word ngram (integer, defaults to 1, using only unigrams)

`thread` integer with the number of threads to use. Defaults to 1.

`...` arguments passed on to ruimtehol:::textspace. See the details below.

Value

an object of class textspace which is a list with elements

- model: a Rcpp pointer to the model
- args: a list with elements
  1. file: the binary file of the model saved on disk
  2. dim: the dimension of the embedding
  3. data: data-specific Starspace training parameters
  4. param: algorithm-specific Starspace training parameters
  5. dictionary: parameters which define ths dictionary of words and labels in Starspace
  6. options: parameters specific to duration of training, the text preparation and the training batch size
  7. test: parameters specific to model testing
- iter: a list with element epoch, lr, error and error_validation showing the error after each epoch

Note

The function `starspace` is a tiny wrapper over the internal function ruimtehol:::textspace which allows direct access to the C++ code in order to run Starspace.
The following arguments are available in that functionality when you do the training. Default settings are shown next to the definition. Some of these arguments are directly set in the `starspace`

function, others can be passed on with ... .

Arguments which define how the training is done:

- dim: size of embedding vectors [100]
- epoch: number of epochs [5]
- lr: learning rate [0.01]
- loss: loss function hinge, softmax [hinge]
- margin: margin parameter in hinge loss. It's only effective if hinge loss is used. [0.05]
- similarity: takes value in [cosine, dot]. Whether to use cosine or dot product as similarity function in hinge loss. It's only effective if hinge loss is used. [cosine]
- negSearchLimit: number of negatives sampled [50]
- maxNegSamples: max number of negatives in a batch update [10]
- p: normalization parameter: normalize sum of embeddings by dividing Size^p [0.5]
- adagrad: whether to use adagrad in training [1]
- ws: only used in trainMode 5, the size of the context window for word level training. [5]
- dropoutLHS: dropout probability for LHS features. [0]
- dropoutRHS: dropout probability for RHS features. [0]
- shareEmb: whether to use the same embedding matrix for LHS and RHS. [1]
- initRandSd: initial values of embeddings are randomly generated from normal distribution with mean=0, standard deviation=initRandSd. [0.001]

Arguments specific to the dictionary of words and labels:

- minCount: minimal number of word occurences [1]
- minCountLabel: minimal number of label occurences [1]
- ngrams: max length of word ngram [1]
- bucket: number of buckets [100000]
- label: labels prefix [__label__]

Arguments which define early stopping or proceeding of model building:

- initModel: if not empty, it loads a previously trained model in -initModel and carry on training.
- validationFile: validation file path
- validationPatience: number of iterations of validation where does not improve before we stop training [10]
- saveEveryEpoch: save intermediate models after each epoch [0]
- saveTempModel: save intermediate models after each epoch with an unique name including epoch number [0]
- maxTrainTime: max train time (secs) [8640000]

Other:

- trainWord: whether to train word level together with other tasks (for multi-tasking). [0]
- wordWeight: if trainWord is true, wordWeight specifies example weight for word level training examples. [0.5]
- useWeight whether input file contains weights [0]

References

https://github.com/facebookresearch

Examples

```
## Not run:
data(dekamer, package = "ruimtehol")
x <- strsplit(dekamer$question, "\\W")
x <- lapply(x, FUN = function(x) x[x != ""])
x <- sapply(x, FUN = function(x) paste(x, collapse = " "))

idx <- sample.int(n = nrow(dekamer), size = round(nrow(dekamer) * 0.7))
writeLines(x[idx], con = "traindata.txt")
writeLines(x[-idx], con = "validationdata.txt")

set.seed(123456789)
m <- starspace(file = "traindata.txt", validationFile = "validationdata.txt",
               trainMode = 5, dim = 10,
               loss = "softmax", lr = 0.01, ngrams = 2, minCount = 5,
               similarity = "cosine", adagrad = TRUE, ws = 7, epoch = 3,
               maxTrainTime = 10)
str(starspace_dictionary(m))
wordvectors <- as.matrix(m)
wv <- starspace_embedding(m,
                          x = c("Nationale Loterij", "migranten", "pensioen"),
                          type = "ngram")
wv
mostsimilar <- embedding_similarity(wordvectors, wv["pensioen", ])
head(sort(mostsimilar[, 1], decreasing = TRUE), 10)
starspace_knn(m, "koning")

## clean up for cran
file.remove(c("traindata.txt", "validationdata.txt"))

## End(Not run)
```

starspace_dictionary *Get the dictionary of a Starspace model*

Description

Get the dictionary of a Starspace model

Usage

```
starspace_dictionary(object)
```

Arguments

object	an object of class textspace as returned by starspace or starspace_load_model

Value

a list with elements

1. ntokens: The number of tokens in the data
2. nwords: The number of words which are part of the dictionary
3. nlabels: The number of labels which are part of the dictionary
4. labels: A character vector with the labels
5. dictionary_size: The size of the dictionary (nwords + nlabels)
6. dictionary: A data.frame with all the words and labels from the dictionary. This data.frame has columns term, is_word and is_label indicating for each term if it is a word or a label

Examples

```
data(dekamer, package = "ruimtehol")
dekamer <- subset(dekamer, depotdat < as.Date("2017-02-01"))
dekamer$text <- strsplit(dekamer$question, "\\W")
dekamer$text <- lapply(dekamer$text, FUN = function(x) x[x != ""])
dekamer$text <- sapply(dekamer$text,
                       FUN = function(x) paste(x, collapse = " "))
dekamer$question_theme_main <- gsub(" ", "-", dekamer$question_theme_main)

set.seed(123456789)
model <- embed_tagspace(x = tolower(dekamer$text),
                        y = dekamer$question_theme_main,
                        early_stopping = 0.8,
                        dim = 10, minCount = 5)
dict <- starspace_dictionary(model)
str(dict)
```

`starspace_embedding` *Get the document or ngram embeddings*

Description

Get the document or ngram embeddings

Usage

```
starspace_embedding(object, x, type = c("document", "ngram"))
```

Arguments

`object`	an object of class `textspace` as returned by `starspace` or `starspace_load_model`
`x`	character vector with text to get the embeddings • If `type` is set to 'document', will assume that a tab or a space is used as separator of each element of `x`.

- If type is set to 'ngram', will assume that a space is used as separator of each element of x.

type the type of embedding requested. Either one of 'document' or 'ngram'. In case of document, the function returns the document embedding, in case of ngram the function returns the embedding of the provided ngram term. See the details section

Details

- document embeddings look to the features (e.g. words) present in x and summate the embeddings of these to get a document embedding and divide this embedding by size^p in case dot similarity is used and the euclidean norm in case cosine similarity is used. Where size is the number of features (e.g. words) in x. If p=1, it's equivalent to taking average of embeddings while when p=0, it's equivalent to taking sum of embeddings. You can set p and similarity in starspace when you train the model.
- for ngram embeddings, starspace is using a hashing trick to find out in which bucket the ngram lies and then retrieves the embedding of that. Note that if you specify ngram, you need to make sure x contains less features (e.g. words) then you've set ngram when you trained your model with starspace.

Value

a matrix of embeddings

Examples

```
data(dekamer, package = "ruimtehol")
dekamer$text <- strsplit(dekamer$question, "\\W")
dekamer$text <- lapply(dekamer$text, FUN = function(x) x[x != ""])
dekamer$text <- sapply(dekamer$text,
                       FUN = function(x) paste(x, collapse = " "))

set.seed(123456789)
model <- embed_tagspace(x = tolower(dekamer$text),
                        y = dekamer$question_theme_main,
                        similarity = "dot",
                        early_stopping = 0.8, ngram = 1, p = 0.5,
                        dim = 10, minCount = 5)
embedding <- starspace_embedding(model, "federale politie", type = "document")
embedding_dictionary <- as.matrix(model)
embedding
colSums(embedding_dictionary[c("federale", "politie"), ]) / 2^0.5

## Not run:
set.seed(123456789)
model <- embed_tagspace(x = tolower(dekamer$text),
                        y = dekamer$question_theme_main,
                        similarity = "cosine",
                        early_stopping = 0.8, ngram = 1,
                        dim = 10, minCount = 5)
embedding <- starspace_embedding(model, "federale politie", type = "document")
```

```
embedding_dictionary <- as.matrix(model)
euclidean_norm <- function(x) sqrt(sum(x^2))
manual <- colSums(embedding_dictionary[c("federale", "politie"), ])
manual / euclidean_norm(manual)
embedding

set.seed(123456789)
model <- embed_tagspace(x = tolower(dekamer$text),
                        y = dekamer$question_theme_main,
                        similarity = "dot",
                        early_stopping = 0.8, ngram = 3, p = 0,
                        dim = 10, minCount = 5, bucket = 1)
starspace_embedding(model, "federale politie", type = "document")
starspace_embedding(model, "federale politie", type = "ngram")

## End(Not run)
```

`starspace_knn` *K-nearest neighbours using a Starspace model*

Description

K-nearest neighbours using a Starspace model

Usage

```
starspace_knn(object, newdata, k = 5, ...)
```

Arguments

`object`	an object of class `textspace` as returned by `starspace` or `starspace_load_model`
`newdata`	a character string of length 1
`k`	integer with the number of nearest neighbours
`...`	not used

Value

a list with elements input and a data.frame called prediction which has columns called label, similarity and rank

starspace_load_model *Load a Starspace model*

Description

Load a Starspace model

Usage

```
starspace_load_model(
  object,
  method = c("ruimtehol", "tsv-data.table", "binary"),
  ...
)
```

Arguments

object	the path to a Starspace model on disk
method	character indicating the method of loading. Possible values are 'ruimtehol', 'binary' and 'tsv-data.table'. Defaults to 'ruimtehol'. • method `'ruimtehol'` loads the model, embeddings and labels which were saved with saveRDS by calling `starspace_save_model` and re-initilises a new Starspace model with the embeddings and the same parameters used to build the model • method `'binary'` loads the embedding which were saved as a as a binary file using the original methods of the Starspace authors - see `starspace_save_model` • method `'tsv-data.table'` loads the embedding which were saved as a tab-delimited flat file using the fast data.table fread function - see `starspace_save_model`
...	further arguments passed on to `starspace` in case of method 'tsv-data.table'

Value

an object of class textspace

See Also

`starspace_save_model`

Examples

```
data(dekamer, package = "ruimtehol")
dekamer$text <- strsplit(dekamer$question, "\\W")
dekamer$text <- lapply(dekamer$text, FUN = function(x) x[x != ""])
dekamer$text <- sapply(dekamer$text,
                       FUN = function(x) paste(x, collapse = " "))

dekamer$target <- as.factor(dekamer$question_theme_main)
```

```
codes <- data.frame(code = seq_along(levels(dekamer$target)),
                    label = levels(dekamer$target), stringsAsFactors = FALSE)
dekamer$target <- as.integer(dekamer$target)
set.seed(123456789)
model <- embed_tagspace(x = dekamer$text,
                        y = dekamer$target,
                        early_stopping = 0.8,
                        dim = 10, minCount = 5)
starspace_save_model(model, file = "textspace.ruimtehol", method = "ruimtehol",
                     labels = codes)
model <- starspace_load_model("textspace.ruimtehol", method = "ruimtehol")


## clean up for cran
file.remove("textspace.ruimtehol")
```

`starspace_save_model` *Save a starspace model as a binary or tab-delimited TSV file*

Description

Save a starspace model as a binary or a tab-delimited TSV file

Usage

```
starspace_save_model(
  object,
  file = "textspace.ruimtehol",
  method = c("ruimtehol", "tsv-data.table", "binary", "tsv-starspace"),
  labels = data.frame(code = character(), label = character(), stringsAsFactors =
    FALSE)
)
```

Arguments

`object` an object of class `textspace` as returned by `starspace` or `starspace_load_model`

`file` character string with the path to the file where to save the model

`method` character indicating the method of saving. Possible values are 'ruimtehol', 'binary', 'tsv-starspace' and 'tsv-data.table'. Defaults to 'ruimtehol'.

- The first method: `'ruimtehol'` saves the R object and the embeddings and optionally the label definitions with saveRDS. This object can be loaded back in with `starspace_load_model`.
- The second method: `'tsv-data.table'` saves the model embeddings as a tab-delimited flat file using the fast data.table fwrite function
- The third method: `'binary'` saves the model as a binary file using the original methods of the Starspace authors

- The fourth method: 'tsv-starspace' saves the model as a tab-delimited flat file using the original methods of the Starspace authors

labels	a data.frame with at least columns code and label which will be saved in case method is set to 'ruimtehol'. This allows to store the mapping between Starspace labels and your own codes alongside the model, where code is your internal code and label is your label.
A new column will be added to this data.frame called label_starspace which combines the Starspace prefix of the label with the code column of your provided data.frame, as this combination is the label starspace uses internally.

Value

invisibly, the character string with the file of the saved object

Note

It is advised to always use method 'ruimtehol' method as it works nicely together with the starspace_load_model function. It is the advised method unless you need to provide non-R users the models and you prefer using the methods provided by the Starspace authors instead of the faster and more portable 'ruimtehol' method.

See Also

starspace_load_model

Examples

```
data(dekamer, package = "ruimtehol")
dekamer$text <- strsplit(dekamer$question, "\\W")
dekamer$text <- lapply(dekamer$text, FUN = function(x) x[x != ""])
dekamer$text <- sapply(dekamer$text,
                       FUN = function(x) paste(x, collapse = " "))

dekamer$target <- as.factor(dekamer$question_theme_main)
codes <- data.frame(code = seq_along(levels(dekamer$target)),
                    label = levels(dekamer$target), stringsAsFactors = FALSE)
dekamer$target <- as.integer(dekamer$target)
set.seed(123456789)
model <- embed_tagspace(x = dekamer$text,
                        y = dekamer$target,
                        early_stopping = 0.8,
                        dim = 10, minCount = 5)
starspace_save_model(model, file = "textspace.ruimtehol", method = "ruimtehol",
                     labels = codes)
model <- starspace_load_model("textspace.ruimtehol", method = "ruimtehol")
starspace_save_model(model, file = "embeddings.tsv", method = "tsv-data.table")

## clean up for cran
file.remove("textspace.ruimtehol")
file.remove("embeddings.tsv")
```

Index