Package 'fastrtext'

January 4, 2018

Type Package

Title 'fastText' Wrapper for Text Classification and Word Representation

Version 0.2.5

Date 2018-01-04

Maintainer Michaël Benesty <michael@benesty.fr>

Description Learning text representations and text classifiers may rely on the same simple and efficient approach. 'fastText' is an open-source, free, lightweight library that allows users to perform both tasks. It transforms text into continuous vectors that can later be used on many language related task. It works on standard, generic hardware (no 'GPU' required). It also includes model size reduction feature. 'fastText' original source code is available at <https://github.com/facebookresearch/fastText>.

URL https://github.com/pommedeterresautee/fastrtext, https://pommedeterresautee.github.io/fastrtext/

BugReports https://github.com/pommedeterresautee/fastrtext/issues

License MIT + file LICENSE

Depends R (>= 3.3)

Imports methods, Rcpp (>= 0.12.12), assertthat

Suggests knitr, testthat

LinkingTo Rcpp

LazyData true

VignetteBuilder knitr

RoxygenNote 6.0.1

Encoding UTF-8

NeedsCompilation yes

Author Michaël Benesty [aut, cre, cph], Facebook, Inc [cph]

Repository CRAN

Date/Publication 2018-01-04 19:50:16 UTC

R topics documented:

`add_tags` *Add tags to documents*

Description

Add tags in the ‘fastText“ format. This format is require for the training step.

Usage

```
add_tags(documents, tags, prefix = "__label__")
```

Arguments

`documents`	texts to learn
`tags`	labels provided as a list or a vector. There can be 1 or more per document.
`prefix`	character to add in front of tag (`fastText` format)

Value

character ready to be written in a file

Examples

```
library(fastrtext)
tags <- list(c(1, 5), 0)
documents <- c("this is a text", "this is another document")
add_tags(documents = documents, tags = tags)
```

---

execute *Execute command on* fastText *model (including training)*

Description

Use the same commands than the one to use for the command line.

Usage

```
execute(commands)
```

Arguments

commands	character of commands

Examples

```
## Not run:
# Supervised learning example
library(fastrtext)

data("train_sentences")
data("test_sentences")

# prepare data
tmp_file_model <- tempfile()

train_labels <- paste0("__label__", train_sentences[,"class.text"])
train_texts <- tolower(train_sentences[,"text"])
train_to_write <- paste(train_labels, train_texts)
train_tmp_file_txt <- tempfile()
writeLines(text = train_to_write, con = train_tmp_file_txt)

test_labels <- paste0("__label__", test_sentences[,"class.text"])
test_texts <- tolower(test_sentences[,"text"])
test_to_write <- paste(test_labels, test_texts)

# learn model
```

```
execute(commands = c("supervised", "-input", train_tmp_file_txt,
                     "-output", tmp_file_model, "-dim", 20, "-lr", 1,
                     "-epoch", 20, "-wordNgrams", 2, "-verbose", 1))

model <- load_model(tmp_file_model)
predict(model, sentences = test_sentences[1, "text"])

# Unsupervised learning example
library(fastrtext)

data("train_sentences")
data("test_sentences")
texts <- tolower(train_sentences[,"text"])
tmp_file_txt <- tempfile()
tmp_file_model <- tempfile()
writeLines(text = texts, con = tmp_file_txt)
execute(commands = c("skipgram", "-input", tmp_file_txt, "-output", tmp_file_model, "-verbose", 1))

model <- load_model(tmp_file_model)
dict <- get_dictionary(model)
get_word_vectors(model, head(dict, 5))

## End(Not run)
```

`fastrtext` *fastrtext: 'fastText' Wrapper for Text Classification and Word Representation*

Description

Learning text representations and text classifiers may rely on the same simple and efficient approach. 'fastText' is an open-source, free, lightweight library that allows users to perform both tasks. It transforms text into continuous vectors that can later be used on many language related task. It works on standard, generic hardware (no 'GPU' required). It also includes model size reduction feature. 'fastText' original source code is available at <https://github.com/facebookresearch/fastText>.

Author(s)

Maintainer: Michaël Benesty <michael@benesty.fr> [copyright holder]

Other contributors:

- Facebook, Inc <bojanowski@fb.com> [copyright holder]

See Also

Useful links:

- https://github.com/pommedeterresautee/fastrtext
- https://pommedeterresautee.github.io/fastrtext/
- Report bugs at https://github.com/pommedeterresautee/fastrtext/issues

get_analogies *Get analogy*

Description

From Mikolov paper Based on related move of a vector regarding a basis. King is to Quenn what a man is to ??? w1 - w2 + w3

Usage

```
get_analogies(model, w1, w2, w3, k = 1)
```

Arguments

model	trained fastText model. NULL if train a new model.
w1	1st word, basis
w2	2nd word, move
w3	3d word, new basis
k	number of words to return

Value

a numeric with distances and names are words

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_unsupervised_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
get_analogies(model, "experience", "experiences", "result")
```

get_dictionary *Get list of known words*

Description

Get a character containing each word seen during training.

Usage

```
get_dictionary(model)
```

Arguments

model	trained fastText model

Value

character containing each word

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_classification_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
print(head(get_dictionary(model), 5))
```

---

get_hamming_loss *Hamming loss*

Description

Compute the hamming loss. When there is only one category, this measure the accuracy.

Usage

```
get_hamming_loss(labels, predictions)
```

Arguments

labels	list of labels
predictions	list returned by the predict command (including both the probability and the categories)

Value

a scalar with the loss

Examples

```
library(fastrtext)
data("test_sentences")
model_test_path <- system.file("extdata", "model_classification_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
sentences <- test_sentences[, "text"]
test_labels <- test_sentences[, "class.text"]
predictions <- predict(model, sentences)
get_hamming_loss(as.list(test_labels), predictions)
```

get_labels *Get list of labels (supervised model)*

Description

Get a character containing each label seen during training.

Usage

```
get_labels(model)
```

Arguments

model	trained fastText model

Value

character containing each label

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_classification_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
print(head(get_labels(model), 5))
```

get_nn *Get nearest neighbour vectors*

Description

Find the k words with the smallest distance. First execution can be slow because of precomputation. Search is done linearly, if your model is big you may want to use an approximate neighbour algorithm from other R packages (like RcppAnnoy).

Usage

```
get_nn(model, word, k)
```

Arguments

model	trained fastText model. Null if train a new model.
word	reference word
k	integer defining the number of results to return

Value

numeric with distances with names as words

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_unsupervised_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
get_nn(model, "time", 10)
```

`get_parameters` *Export hyper parameters*

Description

Retrieve hyper parameters used to train the model

Usage

```
get_parameters(model)
```

Arguments

`model`	trained `fastText` model

Value

list containing each parameter

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_classification_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
print(head(get_parameters(model), 5))
```

get_sentence_representation *Get sentence embedding*

Description

Sentence is splitted in words (using space characters), and word embeddings are averaged.

Usage

```
get_sentence_representation(model, sentences)
```

Arguments

model	fastText model
sentences	character containing the sentences

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_unsupervised_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
m <- get_sentence_representation(model, "this is a test")
print(m)
```

get_tokenized_text *Tokenize text*

Description

Separate words in a text using space characters

Usage

```
get_tokenized_text(model, texts)
```

Arguments

model	fastText model
texts	a character containing the documents

Value

a list of character containing words

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_unsupervised_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
tokens <- get_tokenized_text(model, "this is a test")
print(tokens)
tokens <- get_tokenized_text(model, c("this is a test 1", "this is a second test!"))
print(tokens)
```

---

get_word_distance *Distance between two words*

Description

Distance is equal to `1 - cosine`

Usage

```
get_word_distance(model, w1, w2)
```

Arguments

`model`	trained `fastText` model. Null if train a new model.
`w1`	first word to compare
`w2`	second word to compare

Value

a `scalar` with the distance

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_unsupervised_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
get_word_distance(model, "time", "timing")
```

get_word_ids *Retrieve word IDs*

Description

Get ID of words in the dictionary

Usage

```
get_word_ids(model, words)
```

Arguments

model	fastText model
words	character containing words to retrieve IDs

Value

numeric of ids

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_unsupervised_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
ids <- get_word_ids(model, c("this", "is", "a", "test"))

# print positions
print(ids)
# retrieve words in the dictionary using the positions retrieved
print(get_dictionary(model)[ids])
```

get_word_vectors *Get word embeddings*

Description

Return the vector representation of provided words (unsupervised training) or provided labels (supervised training).

Usage

```
get_word_vectors(model, words = get_dictionary(model))
```

Arguments

model	trained fastText model
words	character of words. Default: return every word from the dictionary.

Value

matrix containing each word embedding as a row and rownames are populated with word strings.

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_unsupervised_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
get_word_vectors(model, c("introduction", "we"))
```

---

load_model *Load an existing fastText trained model*

Description

Load and return a pointer to an existing model which will be used in other functions of this package.

Usage

```
load_model(path)
```

Arguments

path	path to the existing model

Examples

```
library(fastrtext)
model_test_path <- system.file("extdata", "model_classification_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
```

predict.Rcpp_fastrtext *Get predictions (for supervised model)*

Description

Apply the trained model to new sentences. Average word embeddings and search most similar label vector.

Usage

```
## S3 method for class 'Rcpp_fastrtext'
predict(object, sentences, k = 1, simplify = FALSE,
  unlock_empty_predictions = FALSE, ...)
```

Arguments

object	trained fastText model
sentences	character containing the sentences
k	will return the k most probable labels (default = 1)
simplify	when TRUE and k = 1, function return a (flat) numeric instead of a list
unlock_empty_predictions	logical to avoid crash when some predictions are not provided for some sentences because all their words have not been seen during training. This parameter should only be set to TRUE to debug.
...	not used

Value

list containing for each sentence the probability to be associated with k labels.

Examples

```
library(fastrtext)
data("test_sentences")
model_test_path <- system.file("extdata", "model_classification_test.bin", package = "fastrtext")
model <- load_model(model_test_path)
sentence <- test_sentences[1, "text"]
print(predict(model, sentence))
```

print_help *Print help*

Description

Print command information, mainly to use with `execute()` function.

Usage

```
print_help()
```

Examples

```
## Not run:
print_help()

## End(Not run)
```

Rcpp_fastrtext-class *Rcpp_fastrtext class*

Description

Models are S4 objects with several slots (methods) which can be called that way: model$slot_name()

Slots

`load` Load a model

`predict` Make a prediction

`execute` Execute commands

`get_vectors` Get vectors related to provided words

`get_parameters` Get parameters used to train the model

`get_dictionary` List all words learned

`get_labels` List all labels learned

stop_words_sentences *Stop words list*

Description

List of words that can be safely removed from sentences.

Usage

```
stop_words_sentences
```

Format

Character vector of stop words

Source

https://archive.ics.uci.edu/ml/datasets.html?format=&task=&att=&area=&numAtt=&numIns=&type=text&sort=nameUp&view=table

test_sentences *Sentence corpus - test part*

Description

This corpus contains sentences from the abstract and introduction of 30 scientific articles that have been annotated (i.e. labeled or tagged) according to a modified version of the Argumentative Zones annotation scheme.

Usage

```
test_sentences
```

Format

2 data frame with 3117 rows and 2 variables:

text the sentences as a character vector

class.text the category of the sentence

Details

These 30 scientific articles come from three different domains:

1. PLoS Computational Biology (PLOS)
2. The machine learning repository on arXiv (ARXIV)
3. The psychology journal Judgment and Decision Making (JDM)

There are 10 articles from each domain. In addition to the labeled data, this corpus also contains a corresponding set of unlabeled articles. These unlabeled articles also come from PLOS, ARXIV, and JDM. There are 300 unlabeled articles from each domain (again, only the sentences from the abstract and introduction). These unlabeled articles can be used for unsupervised or semi-supervised approaches to sentence classification which rely on a small set of labeled data and a larger set of unlabeled data.

===== References =====

S. Teufel and M. Moens. Summarizing scientific articles: experiments with relevance and rhetorical status. Computational Linguistics, 28(4):409-445, 2002.

S. Teufel. Argumentative zoning: information extraction from scientific text. PhD thesis, School of Informatics, University of Edinburgh, 1999.

Source

https://archive.ics.uci.edu/ml/datasets.html?format=&task=&att=&area=&numAtt=&numIns=&type=text&sort=nameUp&view=table

---

train_sentences *Sentence corpus - train part*

Description

This corpus contains sentences from the abstract and introduction of 30 scientific articles that have been annotated (i.e. labeled or tagged) according to a modified version of the Argumentative Zones annotation scheme.

Usage

```
train_sentences
```

Format

2 data frame with 3117 rows and 2 variables:

text the sentences as a character vector

class.text the category of the sentence

Details

These 30 scientific articles come from three different domains:

1. PLoS Computational Biology (PLOS)
2. The machine learning repository on arXiv (ARXIV)
3. The psychology journal Judgment and Decision Making (JDM)

There are 10 articles from each domain. In addition to the labeled data, this corpus also contains a corresponding set of unlabeled articles. These unlabeled articles also come from PLOS, ARXIV, and JDM. There are 300 unlabeled articles from each domain (again, only the sentences from the abstract and introduction). These unlabeled articles can be used for unsupervised or semi-supervised approaches to sentence classification which rely on a small set of labeled data and a larger set of unlabeled data.

===== References =====

S. Teufel and M. Moens. Summarizing scientific articles: experiments with relevance and rhetorical status. Computational Linguistics, 28(4):409-445, 2002.

S. Teufel. Argumentative zoning: information extraction from scientific text. PhD thesis, School of Informatics, University of Edinburgh, 1999.

Source

https://archive.ics.uci.edu/ml/datasets.html?format=&task=&att=&area=&numAtt=&numIns=&type=text&sort=nameUp&view=table

Index