Package 'fastText'

July 22, 2025

Type Package

Title Efficient Learning of Word Representations and Sentence Classification

Version 1.0.4

Date 2024-02-17

URL https://github.com/mlampros/fastText

BugReports https://github.com/mlampros/fastText/issues

Description An interface to the 'fastText' <https://github.com/facebookresearch/fastText> library for efficient learning of word representations and sentence classification. The 'fastText' algorithm is explained in detail in (i) ``Enriching Word Vectors with subword Information", Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, 2017, <doi:10.1162/tacl_a_00051>; (ii) ``Bag of Tricks for Efficient Text Classification", Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, 2017, <doi:10.18653/v1/e17-2068>; (iii) ``FastText.zip: Compressing text classification models", Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Herve Jegou, Tomas Mikolov, 2016, <doi:10.48550/arXiv.1612.03651>.

License MIT + file LICENSE

SystemRequirements Generally, fastText builds on modern Mac OS and Linux distributions. Since it uses some C++11 features, it requires a compiler with good C++11 support. These include a (g++-4.7.2 or newer) or a (clang-3.3 or newer).

Encoding UTF-8

Imports Rcpp (>= 1.0.0), ggplot2, grid, utils, glue, data.table, stats

Depends R(>= 3.2.3)

LinkingTo Rcpp

Suggests testthat, covr, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.0

NeedsCompilation yes

Author Lampros Mouselimis [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8024-1546>),
Facebook Inc [cph]

Maintainer Lampros Mouselimis <mouselimislampros@gmail.com>

Repository CRAN

Date/Publication 2024-02-17 18:00:02 UTC

Contents

fasttext_interface 2
language_identification 7
plot_progress_logs 9
printAnalogiesUsage 10
printDumpUsage 10
printNNUsage 11
printPredictUsage 11
printPrintNgramsUsage 12
printPrintSentenceVectorsUsage 13
printPrintWordVectorsUsage 13
printQuantizeUsage 14
printTestLabelUsage 15
printTestUsage 15
printUsage 16
print_parameters 16

Index **18**

`fasttext_interface` *Interface for the fasttext library*

Description

Interface for the fasttext library

Usage

```
fasttext_interface(
  list_params,
  path_output = "",
  MilliSecs = 100,
  path_input = "",
  remove_previous_file = TRUE,
  print_process_time = FALSE
)
```

Arguments

list_params	a list of valid parameters
path_output	a character string specifying the file path where the process-logs (or output in generally) should be saved
MilliSecs	an integer specifying the delay in milliseconds when printing the results to the specified *path_output*
path_input	a character string specifying the path to the input data file
remove_previous_file	a boolean. If TRUE, in case that the *path_output* is not an empty string (""), then an existing file with the same output name will be removed
print_process_time	a boolean. If TRUE then the processing time of the function will be printed out in the R session

Details

This function allows the user to run the various methods included in the fasttext library from within R

The "output" parameter which exists in the named list (see examples section) and is passed to the "list_params" parameter of the "fasttext_interface()" function, is a file path and not a directory name and will actually return two files (a *.vec* and a *.bin*) to the output directory.

Value

a vector of class character that includes the parameters and file paths used as input to the function

References

https://github.com/facebookresearch/fastText

https://github.com/facebookresearch/fastText/blob/master/docs/supervised-tutorial.md

Examples

```
## Not run:

library(fastText)


################################################################################
# If the user intends to run the following examples then he / she must replace     #
# the 'input', 'output', 'path_input', 'path_output', 'model' and 'test_data' file #
# paths depending on where the data are located or should be saved!                #
# ( 'tempdir()' is used here as an example folder )                                #
################################################################################


# ----------------------------------------------
# print information for the Usage of each function [ parameters ]
```

```
# ------------------------------------------------

fastText::printUsage()
fastText::printTestUsage()
fastText::printTestLabelUsage()
fastText::printQuantizeUsage()
fastText::printPrintWordVectorsUsage()
fastText::printPrintSentenceVectorsUsage()
fastText::printPrintNgramsUsage()
fastText::printPredictUsage()
fastText::printNNUsage()
fastText::printDumpUsage()
fastText::printAnalogiesUsage()
fastText::print_parameters(command = "supervised")

# ------------------------------------------------------------------------
# In case that the 'command' is one of 'cbow', 'skipgram' or 'supervised'
# ------------------------------------------------------------------------

list_params = list(command = 'cbow',
                   lr = 0.1,
                   dim = 200,
                   input = file.path(tempdir(), "doc.txt"),
                   output = tempdir(),
                   verbose = 2,
                   thread = 1)

res = fasttext_interface(list_params,
                         path_output = file.path(tempdir(),"model_logs.txt"),
                         MilliSecs = 100)


# ---------------------
# 'supervised' training
# ---------------------

list_params = list(command = 'supervised',
                    lr = 0.1,
                    dim = 200,
                    input = file.path(tempdir(), "cooking.train"),
                    output = file.path(tempdir(), "model_cooking"),
                    verbose = 2,
                    thread = 1)

res = fasttext_interface(list_params,
                         path_output = file.path(tempdir(), 'logs_supervise.txt'),
                         MilliSecs = 5)

# ---------------------------------------
# In case that the 'command' is 'predict'
# ---------------------------------------

list_params = list(command = 'predict',
```

```
                    model = file.path(tempdir(), 'model_cooking.bin'),
                    test_data = file.path(tempdir(), 'cooking.valid'),
                    k = 1,
                    th = 0.0)

res = fasttext_interface(list_params,
                         path_output = file.path(tempdir(), 'predict_valid.txt'))


# ------------------------------------
# In case that the 'command' is 'test'  [ k = 5 , means that precision and recall are at 5 ]
# ------------------------------------

list_params = list(command = 'test',
                    model = file.path(tempdir(), 'model_cooking.bin'),
                    test_data = file.path(tempdir(), 'cooking.valid'),
                    k = 5,
                    th = 0.0)

res = fasttext_interface(list_params)  # It only prints 'Precision', 'Recall' to the R session


# ----------------------------------------
# In case that the 'command' is 'test-label'  [ k = 5 , means that precision and recall are at 5 ]
# ----------------------------------------

list_params = list(command = 'test-label',
                    model = file.path(tempdir(), 'model_cooking.bin'),
                    test_data = file.path(tempdir(), 'cooking.valid'),
                    k = 5,
                    th = 0.0)

res = fasttext_interface(list_params,          # prints also 'Precision', 'Recall' to R session
                         path_output = file.path(tempdir(), "test_valid.txt"))

# -----------------
# quantize function  [ it will take a .bin file and return an .ftz file ]
# -----------------

# the quantize function is currenlty (01/02/2019) single-threaded
# https://github.com/facebookresearch/fastText/issues/353#issuecomment-342501742

list_params = list(command = 'quantize',
                    input = file.path(tempdir(), 'model_cooking.bin'),
                output = file.path(tempdir(), gsub('.bin', '.ftz', 'model_cooking.bin')))

res = fasttext_interface(list_params)


# -----------------
# quantize function  [ by using the optional parameters 'qnorm' and 'qout' ]
# -----------------
```

```
list_params = list(command = 'quantize',
                   input = file.path(tempdir(), 'model_cooking.bin'),
                  output = file.path(tempdir(), gsub('.bin', '.ftz', 'model_cooking.bin')),
                   qnorm = TRUE,
                   qout = TRUE)

res = fasttext_interface(list_params)


# ------------------
# print-word-vectors   [ each line of the 'queries.txt' must be a single word ]
# ------------------

list_params = list(command = 'print-word-vectors',
                   model = file.path(tempdir(), 'model_cooking.bin'))

res = fasttext_interface(list_params,
                         path_input = file.path(tempdir(), 'queries.txt'),
                         path_output = file.path(tempdir(), 'print_vecs_file.txt'))


# ----------------------
# print-sentence-vectors  [ See also the comments in the main.cc file about the input-file ]
# ----------------------

list_params = list(command = 'print-sentence-vectors',
                   model = file.path(tempdir(), 'model_cooking.bin'))

res = fasttext_interface(list_params,
                         path_input = file.path(tempdir(), 'text.txt'),
                         path_output = file.path(tempdir(), 'SENTENCE_VECs.txt'))


# ------------
# print-ngrams       [ print to console or to output-file ]
# ------------

list_params = list(command = 'skipgram', lr = 0.1, dim = 200,
                   input = file.path(tempdir(), "doc.txt"),
                   output = tempdir(), verbose = 2, thread = 1,
                   minn = 2, maxn = 2)

res = fasttext_interface(list_params,
                         path_output = file.path(tempdir(), "ngram_out.txt"),
                         MilliSecs = 5)

list_params = list(command = 'print-ngrams',
                   model = file.path(tempdir(), 'ngram_out.bin'),
                  word = 'word')                        # print n-grams for specific word

res = fasttext_interface(list_params, path_output = "")          # print output to console
res = fasttext_interface(list_params,
                     path_output = file.path(tempdir(), "NGRAMS.txt"))   # output to file
```

```
# -------------
# 'nn' function
# -------------

list_params = list(command = 'nn',
                   model = file.path(tempdir(), 'model_cooking.bin'),
                   k = 20,
                   query_word = 'word')          # a 'query_word' is required

res = fasttext_interface(list_params,
                         path_output = file.path(tempdir(), "nn_output.txt"))


# ---------
# analogies  [ in the output file each analogy-triplet-result is separated with a newline ]
# ---------

list_params = list(command = 'analogies',
                   model = file.path(tempdir(), 'model_cooking.bin'),
                   k = 5)

res = fasttext_interface(list_params,
                         path_input = file.path(tempdir(), 'analogy_queries.txt'),
                         path_output = file.path(tempdir(), 'analogies_output.txt'))

# -------------
# dump function  [ the 'option' param should be one of 'args', 'dict', 'input' or 'output' ]
# -------------

list_params = list(command = 'dump',
                   model = file.path(tempdir(), 'model_cooking.bin'),
                   option = 'args')

res = fasttext_interface(list_params,
                         path_output = file.path(tempdir(), "DUMP.txt"))


## End(Not run)
```

`language_identification` *Language Identification using fastText*

Description

Language Identification using fastText

Usage

```
language_identification(
  input_obj,
  pre_trained_language_model_path,
  k = 1,
  th = 0,
  threads = 1,
  verbose = FALSE
)
```

Arguments

input_obj	either a valid character string to a valid path where each line represents a different text extract or a vector of text extracts
pre_trained_language_model_path	a valid character string to the pre-trained language identification model path, for more info see https://fasttext.cc/docs/en/language-identification.html
k	predict top k labels (1 by default)
th	probability threshold (0.0 by default)
threads	an integer specifying the number of threads to run in parallel. This parameter applies only if k > 1
verbose	if TRUE then information will be printed out in the console

Value

an object of class data.table which includes two or more columns with the names 'iso_lang_N' and 'prob_N' where 'N' corresponds to 1 to 'k' input parameter

References

https://fasttext.cc/docs/en/language-identification.html https://becominghuman.ai/a-handy-pre-trained-model-for-language-identification-cadd89db9db8

Examples

```
library(fastText)

vec_txt = c("Incapaz de distinguir la luna y la cara de esta chica,
            Las estrellas se ponen nerviosas en el cielo",
            "Unable to tell apart the moon and this girl's face,
            Stars are flustered up in the sky.")

file_pretrained = system.file("language_identification/lid.176.ftz", package = "fastText")

dtbl_out = language_identification(input_obj = vec_txt,
                                   pre_trained_language_model_path = file_pretrained,
                                   k = 3,
                                   th = 0.0,
                                   verbose = TRUE)
```

```
dtbl_out
```

`plot_progress_logs` *Plot the progress of loss, learning-rate and word-counts*

Description

Plot the progress of loss, learning-rate and word-counts

Usage

```
plot_progress_logs(path_logs = "progress_data.txt", plot = FALSE)
```

Arguments

`path_logs`	a character string specifying a valid path to a file where the progress-logs are saved
`plot`	a boolean specifying if the loss, learning-rate and word-counts should be plotted

Value

an object of class data.frame that includes the progress logs with columns 'progress', 'words_sec_thread', 'learning_rate' and 'loss'

References

http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/

Examples

```
## Not run:

library(fastText)

#----------------------------------------------------------------
# the 'progress_data.txt' file corresponds to the 'path_output'
# parameter of the 'fasttext_interface()'. Therefore the user has
# to run first the 'fasttext_interface()' function to save the
# 'progress_data.txt' file to the desired folder.
#----------------------------------------------------------------

res = plot_progress_logs(path = file.path(tempdir(), "progress_data.txt"),
                         plot = TRUE)


## End(Not run)
```

printAnalogiesUsage *Print Usage Information when the command equals to 'analogies'*

Description

Print Usage Information when the command equals to 'analogies'

Usage

```
printAnalogiesUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printAnalogiesUsage' function in the R session

Examples

```
library(fastText)

printAnalogiesUsage()
```

printDumpUsage *Print Usage Information when the command equals to 'dump'*

Description

Print Usage Information when the command equals to 'dump'

Usage

```
printDumpUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printDumpUsage' function in the R session

Examples

```
library(fastText)

printDumpUsage()
```

`printNNUsage` *Print Usage Information when the command equals to 'nn'*

Description

Print Usage Information when the command equals to 'nn'

Usage

```
printNNUsage(verbose = TRUE)
```

Arguments

`verbose`	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printNNUsage' function in the R session

Examples

```
library(fastText)

printNNUsage()
```

`printPredictUsage` *Print Usage Information when the command equals to 'predict' or 'predict-prob'*

Description

Print Usage Information when the command equals to 'predict' or 'predict-prob'

Usage

```
printPredictUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printPredictUsage' function in the R session

Examples

```
library(fastText)

printPredictUsage()
```

---

printPrintNgramsUsage *Print Usage Information when the command equals to 'print-ngrams'*

Description

Print Usage Information when the command equals to 'print-ngrams'

Usage

```
printPrintNgramsUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printPrintNgramsUsage' function in the R session

Examples

```
library(fastText)

printPrintNgramsUsage()
```

printPrintSentenceVectorsUsage *Print Usage Information when the command equals to 'print-sentence-vectors'*

Description

Print Usage Information when the command equals to 'print-sentence-vectors'

Usage

```
printPrintSentenceVectorsUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printPrintSentenceVectorsUsage' function in the R session

Examples

```
library(fastText)

printPrintSentenceVectorsUsage()
```

printPrintWordVectorsUsage *Print Usage Information when the command equals to 'print-word-vectors'*

Description

Print Usage Information when the command equals to 'print-word-vectors'

Usage

```
printPrintWordVectorsUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printPrintWordVectorsUsage' function in the R session

Examples

```
library(fastText)

printPrintWordVectorsUsage()
```

`printQuantizeUsage` *Print Usage Information when the command equals to 'quantize'*

Description

Print Usage Information when the command equals to 'quantize'

Usage

```
printQuantizeUsage(verbose = TRUE)
```

Arguments

`verbose`	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printQuantizeUsage' function in the R session

Examples

```
library(fastText)

printQuantizeUsage()
```

printTestLabelUsage *Print Usage Information when the command equals to 'test-label'*

Description

Print Usage Information when the command equals to 'test-label'

Usage

```
printTestLabelUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printTestLabelUsage' function in the R session

Examples

```
library(fastText)

printTestLabelUsage()
```

printTestUsage *Print Usage Information when the command equals to 'test'*

Description

Print Usage Information when the command equals to 'test'

Usage

```
printTestUsage(verbose = TRUE)
```

Arguments

verbose	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printTestUsage' function in the R session

Examples

```
library(fastText)

printTestUsage()
```

`printUsage` *Print Usage Information for all parameters*

Description

Print Usage Information for all parameters

Usage

```
printUsage(verbose = TRUE)
```

Arguments

`verbose`	if TRUE then information will be printed in the console

Value

It does not return a value but only prints the available parameters of the 'printUsage' function in the R session

Examples

```
library(fastText)

printUsage()
```

`print_parameters` *Print the parameters for a specific command*

Description

Print the parameters for a specific command

Usage

```
print_parameters(command = "supervised")
```

Arguments

command	a character string specifying the command for which the parameters should be printed in the R session. It should be one of "skipgram", "cbow", "supervised", "test", "test-label" or "quantize"

Value

It does not return a value but only prints the available parameters in the R session

References

https://github.com/facebookresearch/fastText#full-documentation

https://github.com/facebookresearch/fastText/issues/341#issuecomment-339783130

Examples

```
## Not run:

library(fastText)

print_parameters(command = 'supervised')

## End(Not run)
```

Index

fasttext_interface, 2

language_identification, 7

plot_progress_logs, 9
print_parameters, 16
printAnalogiesUsage, 10
printDumpUsage, 10
printNNUsage, 11
printPredictUsage, 11
printPrintNgramsUsage, 12
printPrintSentenceVectorsUsage, 13
printPrintWordVectorsUsage, 13
printQuantizeUsage, 14
printTestLabelUsage, 15
printTestUsage, 15
printUsage, 16