Package 'fastTextR'

December 9, 2023

Type Package

Title An Interface to the 'fastText' Library

Version 2.1.0

Description An interface to the 'fastText' library
<https://github.com/facebookresearch/fastText>. The package
can be used for text classification and to learn word vectors.
An example how to use 'fastTextR' can be found in the 'README' file.

License BSD_3_clause + file LICENSE

Imports stats, graphics, Rcpp (>= 0.12.4), slam

Suggests knitr, rmarkdown

VignetteBuilder knitr

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/EmilHvitfeldt/fastTextR

BugReports https://github.com/EmilHvitfeldt/fastTextR/issues

NeedsCompilation yes

Author Florian Schwendinger [aut],
Emil Hvitfeldt [aut, cre] (<https://orcid.org/0000-0002-0679-1945>)

Maintainer Emil Hvitfeldt <emilhhvitfeldt@gmail.com>

Repository CRAN

Date/Publication 2023-12-09 00:40:09 UTC

R topics documented:

`fasttext` *Create a New* `FastText` *Model*

Description

Create a new `FastText` model. The available methods are the same as the package functions but with out the prefix `"ft_"` and without the need to provide the model.

Usage

```
fasttext()
```

Examples

```
ft <- fasttext()
```

`ft_analogies` *Get Analogies*

Description

TODO

Usage

```
ft_analogies(model, word_triplets, k = 10L)
```

Arguments

`model`	an object inheriting from `"fasttext"`.
`word_triplets`	a character vector of length string giving the word.
`k`	an integer giving the number of nearest neighbors to be returned.

Value

.

Examples

```
## Not run:
ft_analogies(model, c("berlin", "germany", "france"), k = 6L)

## End(Not run)
```

`ft_control` *Default Control Settings*

Description

A auxiliary function for defining the control variables.

Usage

```
ft_control(
  loss = c("softmax", "hs", "ns"),
  learning_rate = 0.05,
  learn_update = 100L,
  word_vec_size = 100L,
  window_size = 5L,
  epoch = 5L,
  min_count = 5L,
  min_count_label = 0L,
  neg = 5L,
  max_len_ngram = 1L,
  nbuckets = 2000000L,
  min_ngram = 3L,
  max_ngram = 6L,
  nthreads = 1L,
  threshold = 1e-04,
  label = "__label__",
  verbose = 0,
  pretrained_vectors = "",
  output = "",
  save_output = FALSE,
  seed = 0L,
  qnorm = FALSE,
  retrain = FALSE,
  qout = FALSE,
  cutoff = 0L,
  dsub = 2L,
  autotune_validation_file = "",
  autotune_metric = "f1",
  autotune_predictions = 1L,
  autotune_duration = 300L,
  autotune_model_size = ""
)
```

Arguments

`loss`	a character string giving the name of the loss function allowed values are `'softmax'`, `'hs'` and `'ns'`.
`learning_rate`	a numeric giving the learning rate, the default value is `0.05`.
`learn_update`	an integer giving after how many tokens the learning rate should be updated. The default value is `100L`, which means the learning rate is updated every 100 tokens.
`word_vec_size`	an integer giving the length (size) of the word vectors.
`window_size`	an integer giving the size of the context window.
`epoch`	an integer giving the number of epochs.
`min_count`	an integer giving the minimal number of word occurences.
`min_count_label`	and integer giving the minimal number of label occurences.
`neg`	an integer giving how many negatives are sampled (only used if loss is `"ns"`).
`max_len_ngram`	an integer giving the maximum length of ngrams used.
`nbuckets`	an integer giving the number of buckets.
`min_ngram`	an integer giving the minimal ngram length.
`max_ngram`	an integer giving the maximal ngram length.
`nthreads`	an integer giving the number of threads.
`threshold`	a numeric giving the sampling threshold.
`label`	a character string specifying the label prefix (default is `'__label__'`).
`verbose`	an integer giving the verbosity level, the default value is `0L` and shouldn't be changed since Rcpp::Rcout cann't handle the traffic.
`pretrained_vectors`	a character string giving the file path to the pretrained word vectors which are used for the supervised learning.
`output`	a character string giving the output file path.
`save_output`	a logical (default is `FALSE`)
`seed`	an integer
`qnorm`	a logical (default is `FALSE`)
`retrain`	a logical (default is `FALSE`)
`qout`	a logical (default is `FALSE`)
`cutoff`	an integer (default is `0L`)
`dsub`	an integer (default is `2L`)

`autotune_validation_file`	a character string
`autotune_metric`	a character string (default is `"f1"`)
`autotune_predictions`	an integer (default is `1L`)
`autotune_duration`	an integer (default is `300L`)
`autotune_model_size`	a character string

Value

a list with the control variables.

Examples

```
ft_control(learning_rate=0.1)
```

`ft_load` *Load Model*

Description

Load a previously saved model from file.

Usage

```
ft_load(file)
```

Arguments

`file`	a character string giving the name of the file to be read in.

Value

an object inheriting from `"fasttext"`.

Examples

```
## Not run:
model <- ft_load("dbpedia.bin")

## End(Not run)
```

`ft_nearest_neighbors` *Get Nearest Neighbors*

Description

TODO

Usage

```
ft_nearest_neighbors(model, word, k = 10L)
```

Arguments

`model`	an object inheriting from `"fasttext"`.
`word`	a character string giving the word.
`k`	an integer giving the number of nearest neighbors to be returned.

Value

.

Examples

```
## Not run:
ft_nearest_neighbors(model, "enviroment", k = 6L)

## End(Not run)
```

`ft_normalize` *Normalize*

Description

Applies normalization to a given text.

Usage

```
ft_normalize(txt)
```

Arguments

`txt`	a character vector to be normalized.

Value

a character vector.

Examples

```
## Not run:
ft_normalize(some_text)

## End(Not run)
```

`ft_save` *Write Model*

Description

Write a previously saved model from file.

Usage

```
ft_save(model, file, what = c("model", "vectors", "output"))
```

Arguments

`model`	an object inheriting from `'fasttext'`.
`file`	a character string giving the name of the file.
`what`	a character string giving what should be saved.

Examples

```
## Not run:
ft_save(model, "my_model.bin", what = "model")

## End(Not run)
```

`ft_sentence_vectors` *Get Sentence Vectors*

Description

Obtain sentence vectors from a previously trained model.

Usage

```
ft_sentence_vectors(model, sentences)
```

Arguments

`model`	an object inheriting from `"fasttext"`.
`sentences`	a character vector giving the sentences.

Value

a matrix containing the sentence vectors.

Examples

```
## Not run:
ft_sentence_vectors(model, c("sentence", "vector"))

## End(Not run)
```

---

ft_test *Evaluate the Model*

Description

Evaluate the quality of the predictions. For the model evaluation precision and recall are used.

Usage

```
ft_test(model, file, k = 1L, threshold = 0)
```

Arguments

`model`	an object inheriting from `'fasttext'`.
`file`	a character string giving the location of the validation file.
`k`	an integer giving the number of labels to be returned.
`threshold`	a double giving the threshold.

Examples

```
## Not run:
ft_test(model, file)

## End(Not run)
```

ft_train *Train a Model*

Description

Train a new word representation model or supervised classification model.

Usage

```
ft_train(
  file,
  method = c("supervised", "cbow", "skipgram"),
  control = ft_control(),
  ...
)
```

Arguments

`file`	a character string giving the location of the input file.
`method`	a character string giving the method, possible values are `'supervised'`, `'cbow'` and `'skipgram'`.
`control`	a list giving the control variables, for more information see `ft_control`.
`...`	additional control arguments inserted into the control list.

Examples

```
## Not run:
cntrl <- ft_control(nthreads = 1L)
model <- ft_train("my_data.txt", method="supervised", control = cntrl)

## End(Not run)
```

ft_words *Get Words*

Description

Obtain all the words from a previously trained model.

Usage

```
ft_words(model)
```

Arguments

`model`	an object inheriting from `"fasttext"`.

Value

a character vector.

Examples

```
## Not run:
ft_words(model)

## End(Not run)
```

`ft_word_vectors` *Get Word Vectors*

Description

Obtain word vectors from a previously trained model.

Usage

```
ft_word_vectors(model, words)
```

Arguments

`model`	an object inheriting from `"fasttext"`.
`words`	a character vector giving the words.

Value

a matrix containing the word vectors.

Examples

```
## Not run:
ft_word_vectors(model, c("word", "vector"))

## End(Not run)
```

predict.supervised_model

Predict using a Previously Trained Model

Description

Predict values based on a previously trained model.

Usage

```
ft_predict(
  model,
  newdata,
  k = 1L,
  threshold = 0,
  rval = c("sparse", "dense", "slam"),
  ...
)
```

Arguments

`model`	an object inheriting from `'fasttext'`.
`newdata`	a character vector giving the new data.
`k`	an integer giving the number of labels to be returned.
`threshold`	a double withing `[0, 1]` giving lower bound on the probabilities. Predictions with probabilities below this lower bound are not returned. The default is `0` which means all predictions are returned.
`rval`	a character string controlling the return value, allowed values are `"sparse"`, `"dense"` and `"slam"`. The default is sparse, here the values are returned as a `data.frame` in a format similar to a simple triplet matrix (sometimes refereed to as the coordinate format). If `rval` is set to `"dense"`, a matrix of the probabilities is returned. Similarly if `rval` is set to `"slam"`, a matrix in the simple triplet sparse format from the **slam** package is returned.
`...`	currently not used.

Value

`NULL` if a `'result_file'` is given otherwise if `'prob'` is true a `data.frame` with the predicted labels and the corresponding probabilities, if `'prob'` is false a character vector with the predicted labels.

Examples

```
## Not run:
ft_predict(model, newdata)

## End(Not run)
```

Index