Package 'sentiment.ai'

October 14, 2022

Type Package

Title Simple Sentiment Analysis Using Deep Learning

Version 0.1.1

Maintainer Ben Wiseman <benjamin.h.wiseman@gmail.com>

Description Sentiment Analysis via deep learning and gradient boosting models with a lot of the underlying hassle taken care of to make the process as simple as possible.
In addition to out-performing traditional, lexicon-based sentiment analysis (see <https://benwiseman.github.io/sentiment.ai/#Benchmarks>),
it also allows the user to create embedding vectors for text which can be used in other analyses.
GPU acceleration is supported on Windows and Linux.

URL https://benwiseman.github.io/sentiment.ai/,
https://github.com/BenWiseman/sentiment.ai

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

VignetteBuilder knitr

Depends R (>= 4.0.0)

Imports data.table (>= 1.12.8), jsonlite, reticulate (>= 1.16), roperators (>= 1.2.0), stats, tensorflow (>= 2.2.0), tfhub (>= 0.8.0), utils, xgboost

Suggests rmarkdown, knitr, magrittr, microbenchmark, prettydoc, rappdirs, rstudioapi, text2vec (>= 0.6)

Collate 'package-sentiment_ai.R' 'init_and_install.R' 'sentiment.R' 'choose_model.R' 'data-default_data.R' 'data-example_data.R' 'object-sentiment_env.R' 'matrix_helpers.R' 'constants.R' 'create_error_text.R' 'utils-data-table.R' 'globals.R' 'local_from_reticulate.R'

NeedsCompilation no

Author Ben Wiseman [cre, aut, ccp],
Steven Nydick [aut] (<https://orcid.org/0000-0002-2908-1188>),
Tristan Wisner [aut],
Fiona Lodge [ctb],
Yu-Ann Wang [ctb],
Veronica Ge [art],
Korn Ferry Institute [fnd]

Repository CRAN

Date/Publication 2022-03-19 00:00:05 UTC

R topics documented:

`sentiment.ai-package` *sentiment.ai: Simple Sentiment Analysis Using Deep Learning*

Description

This package uses Google's Universal Sentence Encoder, simplifies all the difficulties, and turns it into a sentiment analysis package.

Main Benefits:

- Tolerates spelling mitsakes
- Not dependent on exactly matching a fixed dictionary
- Requires less pre-processing
- More powerful than dictionary-based methods

Main Drawbacks:

- Requires a lot of RAM

- Can be slow on larger datasets (unless using GPU)

Effectively, if you have a reasonably powerful computer, you can use sentiment.ai as a more flexible, powerful, and modern approach to sentiment analysis.

Author(s)

Maintainer: Ben Wiseman <benjamin.h.wiseman@gmail.com> [conceptor]

Authors:

- Steven Nydick <swnydick@gmail.com> (ORCID)
- Tristan Wisner <tristan.wisner@kornferry.com>

Other contributors:

- Fiona Lodge <fiona.lodge@kornferry.com> [contributor]
- Yu-Ann Wang <yu-ann-wang@kornferry.com> [contributor]
- Veronica Ge <veronica.ge@kornferry.com> [artist]
- Korn Ferry Institute <KFInstituteRequests@KornFerry.com> [funder]

See Also

Useful links:

- https://benwiseman.github.io/sentiment.ai/
- https://github.com/BenWiseman/sentiment.ai

---

airline_tweets *Airline Tweet Data*

Description

This is a basic data that can be used to test out the installation and make sure that the model is running correctly. The current data is based on airline tweets with corresponding sentiment labels.

Usage

```
data(airline_tweets)
```

Format

An object of class `data.table` with the "text" column required for text processing and the "sentiment" column for checking positive or negative predictions of the model.

as_py_list — *as py list because R to Python conversion doesn't work with list is of length 1*

Description

as py list because R to Python conversion doesn't work with list is of length 1

Usage

```
as_py_list(x)
```

Arguments

x	character vector that is to be passed into tensorflowtext via reticulate.

Value

List if x is length 1 else x.

default — *Default sentiment matching dictionary*

Description

This is a default sentiment matching dictionary with over 100 pairs of positive:negative examples. Feel free to use as-is or use as an example to create your own. The main point is that there needs to be a corresponding negative for each positive.

Usage

```
data(default)
```

Format

An object of class `"data.table"`

Examples

```
# For Example
pos <- c("good apples", "fresh", "delicious")
neg <- c("bad apples", "not fresh", "not delicious")

# If positive was:
c("good", "fresh", "delicious")

# Then "These were some good apples" would be seen as closer to a negative example!
```

embed_text *Create Text Embedding Matrix*

Description

turns character vector into `length(text) x 512` embedding matrix. For power users. Requires `init_sentiment.ai()` to have been called!

Usage

```
embed_text(text, batch_size = NULL, model = NULL)
```

Arguments

`text`	character vector to be embedded. Note that longer comments take longer.
`batch_size`	integer - how many to embed at once. Higher numbers are faster but use more memory.
`model`	character - the embedding model to use (same as `sentiment_score()`).

Value

numeric matrix of `length(text) x 512`. Original text is stored in the row names attribute.

get_default_embedding *get default embedding If it exists, return the object. If not, try downloading it. If download works, return object. Else return* `NULL` *(to be handles in* `embed_topics()`*).*

Description

get default embedding If it exists, return the object. If not, try downloading it. If download works, return object. Else return `NULL` (to be handles in `embed_topics()`).

Usage

```
get_default_embedding(model)
```

Arguments

`model`	character - whic model's default embeddings are needed

`install_default_embeddings` *Function to grab the default embeddings for* `sentiment_match()` *Necessary to keep package size under 5Mb. Will check if they're there, if so return TRUE. If they are not there, try download and return TRUE. Otherwise, return FALSE (and generate them - will take a few seconds!).*

Description

Function to grab the default embeddings for `sentiment_match()` Necessary to keep package size under 5Mb. Will check if they're there, if so return TRUE. If they are not there, try download and return TRUE. Otherwise, return FALSE (and generate them - will take a few seconds!).

Usage

```
install_default_embeddings()
```

`install_scoring_model` *Install a Scoring Model*

Description

Install a Scoring Model

Usage

```
install_scoring_model(
  model = c("en.large", "en", "multi.large", "multi"),
  scoring = c("xgb", "glm"),
  scoring_version = "1.0",
  ...
)
```

Arguments

`model`	The embedding model, one of c("en.large", "en", "multi.large", "multi").
`scoring`	The scoring model, currently one of: • "xgb" does default xgboost • "glm" does generalized linear model (if you can't run xgboost)
`scoring_version`	Version of scoring model (will add more over time)
`...`	Additional options to the function, including: • repo_url: OPTIONAL custom github repo blob url for external scoring models. The default repo_url is "https://github.com/BenWiseman/sentiment.ai/blob/main/models"

Details

This downloads the scoring models from a set repository in order to keep the main package within CRAN size limits.

In the future, this will also make it possible for the community to add new and improved models!

Value

0 if model did not need to be downloaded. 1 if model needed to be downloaded.

`install_sentiment.ai` *setup*

Description

Install and Setup sentiment.ai package

Usage

```
install_sentiment.ai(
  envname = "r-sentiment-ai",
  method = c("auto", "virtualenv", "conda"),
  gpu = FALSE,
  python_version = "3.8.10",
 modules = list(numpy = "1.19.5", sentencepiece = "0.1.95", tensorflow = "2.4.1",
    tensorflow_hub = "0.12.0", `tensorflow-text` = "2.4.3"),
  fresh_install = TRUE,
  restart_session = TRUE,
  ...
)

init_sentiment.ai(
  model = c("en.large", "multi.large", "en", "multi"),
  envname = "r-sentiment-ai",
  method = c("auto", "virtualenv", "conda"),
  silent = FALSE
)

check_sentiment.ai(...)
```

Arguments

`envname`	The name, or full path, of the environment in which Python packages are to be installed. When `NULL` (the default), the active environment as set by the `RETICULATE_PYTHON_ENV` variable will be used; if that is unset, then the `r-reticulate` environment will be used.

method	Installation method. By default, "auto" automatically finds a method that will work in the local environment. Change the default to force a specific installation method. Note that the "virtualenv" method may not available on Windows due to a tensorflow issue. Note also that since this command runs without privilege the "system" method is available only on Windows.
gpu	Whether GPU should be enabled when installing TensorFlow
python_version	The requested Python version. Ignored when attempting to install with a Python virtual environment.
modules	A list of modules needed for installing tensorflow. See details for more information. Only change this argument if you know what you are doing!
fresh_install	Whether to create the Python environment prior to installing the modules or to install everything in an existing environment (if one exists). Only change this argument if you know what you are doing! If the environment does not already exist, will create the environment first.
restart_session	Whether to restart the R session after finishing installation. Only works on Rstudio.
...	Additional arguments passed to conda_install() or virtualenv_install().
model	path to tensorflow hub embedding model. default is both universal sentence encoder en (default) and multi.
silent	logical - do you want to suppress console logging? Can't affect tensorflow/GPU/python/c++ output, unfortunately.

Details

Sets up environment specific for sentiment.ai. The packages that it currently needs are as follows:

Module	Version
python	3.8.10
numpy	1.19.5
tensorflow	2.4.1
tensorflow_hub	0.12.0
tensorflow-text	2.4.3
sentencepiece	0.1.95

Please do not change these unless you know what you are doing.

Note that it installs with like `tensorflow::install_tensorflow` and `pip = TRUE`

Value

NULL this function simply installs the required python dependencies and default scoring models and pre-calculated embedding vectors.

python function to embed text can be returned, but is not necessary. `embed_text()` does this for you.

NULL this function checks if `init_sentiment.ai()` has been called successfully, if not, it is called.

Note

Setting environments with `reticulate` is notoriously difficult. If the RETICULATE_PYTHON environment is set, then reticulate will not let you change the Python binary used (or the Python environment) using `use_condaenv` **or** `use_virtualenv`. This environment can be accidentally set in the following ways:

1. If RETICULATE_PYTHON is in your .Renviron file or bash/zsh rc files. This is the most obvious place that this environment will be set.
2. Using Project Options or Global Options under "Python>Python Interpreter". If this is set, then reticulate will almost always use this version of Python and will not let you change.
3. If you have already loaded reticulate and have run `py_config`. Once a Python version/environment is instantiated, you will not be able to change it and will have to restart R.
4. If you are in **any** project, at all! Currently (as of `reticulate` version 1.22), every project automatically sets the RETICULATE_PYTHON environment variable, either through the Global or Project Options or by using heuristics. If you are in an RStudio project, you **must** update Global/Project Options with the specific version/environment of Python that you want to use, or you will not be able to change it!

Manually setting the environment variable to NULL (using `Sys.unsetenv("RETICULATE_PYTHON")`, updating your Project/Global options going Tools>Project Options or Tools>Global Options and then select Python in the left menu bar and click the "Select" button to manually set the Python interpreter, and/or restarting your R session **might** fix the problem.

We know this is a pain, and we would like to fix this for you, but we are dependent on the RStudio/reticulate team to update how they determine the allowable Python versions/environments.

Examples

```
## Not run:
install_sentiment.ai(envname = "r-sentiment-ai",
                     method  = "conda",
                     python_version = "3.8.10")
init_sentiment.ai(model   = "en.large",
                  envname = "r-sentiment-ai")
check_sentiment.ai()

# if you run into an issue, follow the instructions/see the note and retry!

## End(Not run)
```

`matrix_similarity` *Cosine Similarity*

Description

Cosine Similarity

cosine_match()

Usage

```
cosine(x, y = NULL)

cosine_match(target, reference, keep_target_order = FALSE)
```

Arguments

`x`	A numeric vector or matrix
`y`	A numeric vector or matrix of the same dimensions as x
`target`	numeric matrix of j values where each row is one observation. Use row names as ID.
`reference`	numeric matrix of j values where each row is one observation. Use row names as ID.
`keep_target_order`	logical include column indicating original row order of target matrix.

Value

data.table containing ranked (1 = top) pairwise similarities between target and reference

Examples

```
## Not run:
n <- 5
y <- matrix(rnorm(n * 512), ncol = 512)
x <- matrix(rnorm(n * 512), ncol = 512)

all.equal(cosine(x, y),
          text2vec::sim2(x, y))

## End(Not run)
```

read_embedding *Read embedding file. Take json path, return single embedding object for specific model.*

Description

Read embedding file. Take json path, return single embedding object for specific model.

Usage

```
read_embedding(file, model = "en.large", version = NULL)
```

Arguments

file	character - the filepath to the json object.
model	character - which model's default embeddings are needed.
version	PLACEHOLDER - may be necessary in future.

sentiment.env *Sentiment AI Embedding Environment*

Description

This is the embedding envirnoment for the sentiment.ai model. On package load, this object should be NULL. When the model is initialized, this should be updated based on the required embedding model. The embedding function will be stored in the "f" slot of this environment.

Usage

```
sentiment.env
```

Format

An object of class `environment` of length 0.

sentiment_match *Sentiment Matching*

Description

Provides score and explanation, returns a single vector, and runs relatively fast.

Usage

```
sentiment_match(
  x = NULL,
  phrases = NULL,
  model = names(default_models),
  batch_size = 100,
  ...
)
```

Arguments

x	A plain text vector or column name if data is supplied. If you know what you're doing, you can also pass in a 512-D numeric embedding.
phrases	A named list of examples phrases with each element of the list being words/terms that are indications of the name of that element (such as positive words/terms under the name "positive" and negative words/terms under the name "negative", all within the same list).
model	An embedding name from tensorflow-hub, some of which are "en" (english large or not) and "multi" (multi-lingual large or not).
batch_size	Size of batches to use. Larger numbers will be faster than smaller numbers, but do not exhaust your system memory!
...	Additional arguments passed to conda_install() or virtualenv_install().

Value

data.table containing text, sentiment, phrase, class, and similarity.

Examples

```
## Not run:
envname   <- "r-sentiment-ai"

# make sure to install sentiment ai (install_sentiment.ai)
# install_sentiment.ai(envname = envname,
#                      method  = "conda")

# running the model
mod_match <- sentiment_match(x       = airline_tweets$text,
                             model   = "en.large",
```

```
                              envname = envname)

# checking performance
pos_neg <- factor(airline_tweets$airline_sentiment,
                  levels = c("negative", "neutral", "positive"))
pos_neg <- (as.numeric(pos_neg) - 1) / 2
cosine(mod_match$sentiment, pos_neg)

# you could also calculate accuracy/kappa

## End(Not run)
```

`sentiment_score` *Simple Sentiment Scores*

Description

This uses a simple model (xgboost or glm) to return a simple predictive score, where numbers closer to 1 are more positive and numbers closer to -1 are more negative. This can be used to determine whether the sentiment is positive or negative.

Usage

```
sentiment_score(
  x = NULL,
  model = names(default_models),
  scoring = c("xgb", "glm"),
  scoring_version = "1.0",
  batch_size = 100,
  ...
)
```

Arguments

`x`	A plain text vector or column name if data is supplied. If you know what you're doing, you can also pass in a 512-D numeric embedding.
`model`	An embedding name from tensorflow-hub, some of which are "en" (english large or not) and "multi" (multi-lingual large or not).
`scoring`	Model to use for scoring the embedding matrix (currently either "xgb" or "glm").
`scoring_version`	The scoring version to use, currently only 1.0, but other versions might be supported in the future.
`batch_size`	Size of batches to use. Larger numbers will be faster than smaller numbers, but do not exhaust your system memory!
`...`	Additional arguments passed to `conda_install()` or `virtualenv_install()`.

Details

Uses simple preditive models on embeddings to provide probability of positive score (rescaled to -1:1 for consistency with other packages).

Value

numeric vector of length(x) containing a re-scaled sentiment probabilities.

Examples

```
## Not run:
envname <- "r-sentiment-ai"

# make sure to install sentiment ai (install_sentiment.ai)
# install_sentiment.ai(envname = envname,
#                      method  = "conda")

# running the model
mod_xgb <- sentiment_score(x       = airline_tweets$text,
                           model   = "en.large",
                           scoring = "xgb",
                           envname = envname)
mod_glm <- sentiment_score(x       = airline_tweets$text,
                           model   = "en.large",
                           scoring = "glm",
                           envname = envname)

# checking performance
pos_neg <- factor(airline_tweets$airline_sentiment,
                  levels = c("negative", "neutral", "positive"))
pos_neg <- (as.numeric(pos_neg) - 1) / 2
cosine(mod_xgb, pos_neg)
cosine(mod_glm, pos_neg)

# you could also calculate accuracy/kappa



## End(Not run)
```

Index