Package ‘spacyr’

December 8, 2023

Type Package
Title Wrapper to the 'spaCy' 'NLP' Library
Version 1.3.0
Description An R wrapper to the 'Python' 'spaCy' 'NLP' library, from <https://spacy.io>.
License GPL-3
LazyData TRUE
Depends R (>= 3.0.0), methods
Imports data.table, reticulate (>= 1.6)
Suggests dplyr, knitr, quanteda, R.rsp, rmarkdown, spelling, testthat, tidytext, tibble
URL https://spacyr.quanteda.io
Encoding UTF-8
BugReports https://github.com/quanteda/spacyr/issues
RoxygenNote 7.2.3
Language en-GB
VignetteBuilder R.rsp
NeedsCompilation no
Author Kenneth Benoit [cre, aut, cph]  
( <https://orcid.org/0000-0002-0797-564X>),  
Akitaka Matsuo [aut] (<https://orcid.org/0000-0002-3323-6330>),  
Johannes Gruber [ctb] (<https://orcid.org/0000-0001-9177-1772>),  
European Research Council [fnd] (ERC-2011-StG 283794-QUANTESS)
Maintainer Kenneth Benoit <kbenoit@lse.ac.uk>
Repository CRAN
Date/Publication 2023-12-08 15:20:02 UTC
R topics documented:

spacyr-package .......................................................... 2
data_char_paragraph .................................................... 3
data_char_sentences .................................................... 3
text_extract ............................................................... 3
nounphrase_extract ...................................................... 4
spacy_download_langmodel ............................................. 6
spacy_download_langmodel_virtualenv ............................... 6
spacy_extract_entity ................................................... 7
spacy_extract_nounphrases .......................................... 8
spacy_finalise ............................................................ 9
spacy_initialise .......................................................... 9
spacy_install ............................................................ 10
spacy_install_virtualenv .............................................. 12
spacy_parse .............................................................. 12
spacy_tokenize ........................................................... 14
spacy_uninstall .......................................................... 15
spacy_upgrade ............................................................ 16

Index 18

spacyr-package An R wrapper to the spaCy NLP system

Description

An R wrapper to the Python (Cython) spaCy NLP system, from https://spacy.io. Nicely integrated with quanteda. spacyr is designed to provide easy access to the powerful functionality of spaCy, in a simple format.

Author(s)

Ken Benoit and Akitaka Matsuo

References


See Also

Useful links:

- https://spacy.quanteda.io
- Report bugs at https://github.com/quanteda/spacyr/issues
data_char_paragraph

Description
A sample of text from the Irish budget debate of 2010 (531 tokens long).

Usage
data_char_paragraph

Format
An object of class character of length 1.

data_char_sentences

Description
A character object consisting of 30 short documents in plain text format for testing. Each document is one or two brief sentences.

Usage
data_char_sentences

Format
An object of class character of length 30.

entity_extract

Description
From an object parsed by spacy_parse(), extract the entities as a separate object, or convert the multi-word entities into single "token" consisting of the concatenated elements of the multi-word entities.

Usage
eentity_extract(x, type = c("named", "extended", "all"), concatenator = ".")
eentity_consolidate(x, concatenator = ".")
nounphrase_extract

Arguments

x output from `spacy_parse()`.

type type of named entities, either named, extended, or all. See [https://spacy.io/docs/usage/entity-recognition#entity-types](https://spacy.io/docs/usage/entity-recognition#entity-types) for details.

concatenator the character(s) used to join the elements of multi-word named entities

Value

`entity_extract()` returns a data.frame of all named entities, containing the following fields:

- doc_id name of the document containing the entity
- sentence_id the sentence ID containing the entity, within the document
- entity the named entity
- entity_type the type of named entities (e.g. PERSON, ORG, PERCENT, etc.)

`entity_consolidate` returns a modified data.frame of parsed results, where the named entities have been combined into a single "token". Currently, dependency parsing is removed when this consolidation occurs.

Examples

```r
## Not run:
spacy_initialize()

# entity extraction
txt <- "Mr. Smith of moved to San Francisco in December."
parsed <- spacy_parse(txt, entity = TRUE)
entity_extract(parsed)
entity_extract(parsed, type = "all")

## End(Not run)

## Not run:
# consolidating multi-word entities
txt <- "The House of Representatives voted to suspend aid to South Dakota."
parsed <- spacy_parse(txt, entity = TRUE)
entity_consolidate(parsed)

## End(Not run)
```

nounphrase_extract  Extract or consolidate noun phrases from parsed documents

Description

From an object parsed by `spacy_parse()`, extract the multi-word noun phrases as a separate object, or convert the multi-word noun phrases into single "token" consisting of the concatenated elements of the multi-word noun phrases.
Usage

nounphrase_extract(x, concatenator = "_")
	nounphrase_consolidate(x, concatenator = "_")

Arguments

x 
output from spacy_parse()
concatenator the character(s) used to join elements of multi-word noun phrases

Value

noun returns a data.frame of all named entities, containing the following fields:

- doc_id name of the document containing the noun phrase
- sentence_id the sentence ID containing the noun phrase, within the document
- nounphrase the noun phrase
- root the root token of the noun phrase

nounphrase_consolidate returns a modified data.frame of parsed results, where the noun phrases have been combined into a single "token". Currently, dependency parsing is removed when this consolidation occurs.

Examples

```r
## Not run:
spacy_initialize()
# entity extraction
txt <- "Mr. Smith of moved to San Francisco in December."
parsed <- spacy_parse(txt, nounphrase = TRUE)
entity_extract(parsed)
## End(Not run)
## Not run:
# consolidating multi-word noun phrases
txt <- "The House of Representatives voted to suspend aid to South Dakota."
parsed <- spacy_parse(txt, nounphrase = TRUE)
nounphrase_consolidate(parsed)
## End(Not run)
```
**spacy_download_langmodel**

*Download spaCy language models*

**Description**

Download spaCy language models

**Usage**

```r
spacy_download_langmodel(lang_models = "en_core_web_sm", force = FALSE)
```

**Arguments**

- `lang_models` character; language models to be installed. Defaults `en_core_web_sm` (English model). A vector of multiple model names can be used (e.g. `c("en_core_web_sm", "de_core_news_sm")`). A list of available language models and their names is available from the spaCy language models page.

- `force` ignore if spaCy/the lang_models is already present and install it anyway.

**Value**

Invisibly returns the installation log.

**Examples**

```r
## Not run:
# install medium sized model
spacy_download_langmodel("en_core_web_md")

# install several models with spaCy
spacy_install(lang_models = c("en_core_web_sm", "de_core_news_sm"))

# install transformer based model
spacy_download_langmodel("en_core_web_trf")

## End(Not run)
```

**spacy_download_langmodel_virtualenv**

*Install a language model in a conda or virtual environment*

**Description**

Deprecated. spacyr now always uses a virtual environment, making this function redundant.
Usage

spacy_download_langmodel_virtualenv(...)

Arguments

... not used

spacy_extract_entity  Extract named entities from texts using spaCy

Description

This function extracts named entities from texts, based on the entity tag ent attributes of documents objects parsed by spaCy (see https://spacy.io/usage/linguistic-features#section-named-entities).

Usage

spacy_extract_entity(
  x,
  output = c("data.frame", "list"),
  type = c("all", "named", "extended"),
  multithread = TRUE,
  ...
)

Arguments

x a character object or a TIF-compliant corpus data.frame (see https://github.com/ropenscilabs/tif)
output type of returned object, either "list" or "data.frame".
type type of named entities, either named, extended, or all. See https://spacy.io/docs/usage/entity-recognition/entity-types for details.
multithread logical; If TRUE, the processing is parallelized using spaCy’s architecture (https://spacy.io/api)
... unused

Details

When the option output = "data.frame" is selected, the function returns a data.frame with the following fields.

text contents of entity
entity_type type of entity (e.g. ORG for organizations)
start_id serial number ID of starting token. This number corresponds with the number of data.frame returned from spacy_tokenize(x) with default options.
length number of words (tokens) included in a named entity (e.g. for an entity, "New York Stock Exchange", length = 4)
spacy_extract_nounphrases

Extract noun phrases from texts using spaCy

Description

This function extracts noun phrases from documents, based on the noun_chunks attributes of documents objects parsed by spaCy (see https://spacy.io/usage/linguistic-features#noun-chunks).

Usage

spacy_extract_nounphrases(
  x,
  output = c("data.frame", "list"),
  multithread = TRUE,
  ...
)

Arguments

x a character object or a TIF-compliant corpus data.frame (see https://github.com/ropenscilabs/tif)
output type of returned object, either "data.frame" or "list"
multithread logical; If TRUE, the processing is parallelized using spaCy's architecture (https://spacy.io/api)
... unused

Examples

## Not run:
spacy_initialize()

txt <- c(doc1 = "The Supreme Court is located in Washington D.C.",
         doc2 = "Paul earned a postgraduate degree from MIT.")
spacy_extract_entity(txt)
spacy_extract_entity(txt, output = "list")
## End(Not run)
**Details**

When the option `output = "data.frame"` is selected, the function returns a `data.frame` with the following fields.

- **text**: contents of noun-phrase
- **root_text**: contents of root token
- **start_id**: serial number ID of starting token. This number corresponds with the number of `data.frame` returned from `spacy_tokenize(x)` with default options.
- **root_id**: serial number ID of root token
- **length**: number of words (tokens) included in a noun-phrase (e.g. for a noun-phrase, "individual car owners", `length = 3`)

**Value**

either a list or `data.frame` of tokens

**Examples**

```r
## Not run:
spacy_initialize()

txt <- c(doc1 = "Natural language processing is a branch of computer science.",
         doc2 = "Paul earned a postgraduate degree from MIT.")
spacy_extract_nounphrases(txt)
spacy_extract_nounphrases(txt, output = "list")

## End(Not run)
```

---

**spacy_finalize**

*Finalize spaCy*

**Description**

While running spaCy on Python through R, a Python process is always running in the background and Rsession will take up a lot of memory (typically over 1.5GB). `spacy_finalize()` terminates the Python process and frees up the memory it was using.

**Usage**

`spacy_finalize()`

**Author(s)**

Akitaka Matsuo
spacy_initialize    Initialize spaCy

Description

Initialize spaCy to call from R.

Usage

spacy_initialize(model = "en_core_web_sm", entity = TRUE, ...)

Arguments

model    Language package for loading spaCy. Example: en_core_web_sm (English) and de_core_web_sm (German). Default is en_core_web_sm.
entity    logical: if FALSE is selected, named entity recognition is turned off in spaCy. This will speed up the parsing as it will exclude ner from the pipeline. For details of spaCy pipeline, see https://spacy.io/usage/processing-pipelines. The option FALSE is available only for spaCy version 2.0.0 or higher.
...

Author(s)

Akitaka Matsuo, Johannes B. Gruber

spacy_install    Install spaCy in conda or virtualenv environment

Description

Install spaCy in a self-contained environment, including specified language models.

Usage

spacy_install(
    version = "latest",
    lang_models = "en_core_web_sm",
    ask = interactive(),
    force = FALSE,
    ...
)
Arguments

version character; spaCy version to install (see details).
lang_models character; language models to be installed. Defaults en_core_web_sm (English model). A vector of multiple model names can be used e.g. c("en_core_web_sm", "de_core_news_sm"). A list of available language models and their names is available from the spaCy language models page.
ask logical; ask whether to proceed during the installation. By default, questions are only asked in interactive sessions.
force ignore if spaCy/the lang_models is already present and install it anyway.
...
not used.

Details

The function checks whether a suitable installation of Python is present on the system and installs one via reticulate::install_python() otherwise. It then creates a virtual environment with the necessary packages in the default location chosen by reticulate::virtualenv_root().

If you want to install a different version of Python than the default, you should call reticulate::install_python() directly. If you want to create or use a different virtual environment, you can use, e.g., Sys.setenv(SPACY_PYTHON = "path/to/directory").

See Also

spacy_download_langmodel()

Examples

## Not run:
# install the latest version of spaCy
spacy_install()

# update spaCy
spacy_install(force = TRUE)

# install an older version
spacy_install(version = "3.1.0")

# install with GPU enabled
spacy_install(version = "cuda-autodetect")

# install on Apple ARM processors
spacy_install(version = "apple")

# install an old custom version
spacy_install(version = ["cuda-autodetect"]==3.2.0)

# install several models with spaCy
spacy_install(lang_models = c("en_core_web_sm", "de_core_news_sm"))
# install spaCy to an existing virtual environment
Sys.setenv(RETICULATE_PYTHON = "path/to/python")
spacy_install()

## End(Not run)

spacy_install_virtualenv

*Install spaCy to a virtual environment*

**Description**

Deprecated. `spacy_install` now installs to a virtual environment by default.

**Usage**

spacy_install_virtualenv(...)  

**Arguments**

... not used

spacy_parse

*Parse a text using spaCy*

**Description**

The `spacy_parse()` function calls spaCy to both tokenize and tag the texts, and returns a data.table of the results. The function provides options on the types of tags (tagset_options) either "google" or "detailed", as well as lemmatization (lemma). It provides a functionalities of dependency parsing and named entity recognition as an option. If "full_parse = TRUE" is provided, the function returns the most extensive list of the parsing results from spaCy.

**Usage**

spacy_parse(  
  x,  
  pos = TRUE,  
  tag = FALSE,  
  lemma = TRUE,  
  entity = TRUE,  
  dependency = FALSE,  
  nounphrase = FALSE,  
  multithread = TRUE,  
  additional_attributes = NULL,  
  ...
  )
Arguments

- **x**: A character object, a quantified corpus, or a TIF-compliant corpus data.frame (see https://github.com/ropenscilabs/tif).
- **pos**: Logical whether to return universal dependency POS tagset https://universaldependencies.org/u/pos/.
- **tag**: Logical whether to return detailed part-of-speech tags, for the language model en, it uses the OntoNotes 5 version of the Penn Treebank tag set (https://spacy.io/docs/usage/pos-tagging#pos-schemes). Annotation specifications for other available languages are available on the spaCy website (https://spacy.io/api/annotation).
- **lemma**: Logical; include lemmatized tokens in the output (lemmatization may not work properly for non-English models).
- **entity**: Logical; if TRUE, report named entities.
- **dependency**: Logical; if TRUE, analyse and tag dependencies.
- **nounphrase**: Logical; if TRUE, analyse and tag noun phrases tags.
- **multithread**: Logical; If TRUE, the processing is parallelized using spaCy’s architecture (https://spacy.io/api).
- **additional_attributes**: A character vector; this option is for extracting additional attributes of tokens from spaCy. When the names of attributes are supplied, the output data.frame will contain additional variables corresponding to the names of the attributes. For instance, when additional_attributes = c("is_punct"), the output will include an additional variable named is_punct, which is a Boolean (in R, logical) variable indicating whether the token is a punctuation. A full list of available attributes is available from https://spacy.io/api/token#attributes.
- **...**: Not used directly.

Value

A data.frame of tokenized, parsed, and annotated tokens.

Examples

```r
## Not run:
spacy_initialize()
txt <- "And now for something completely different."
spacy_parse(txt)
spacy_parse(txt, pos = TRUE, tag = TRUE)
spacy_parse(txt, dependency = TRUE)

# Using a list of text documents

txt2 <- c(doc1 = "The fast cat catches mice.\nThe quick brown dog jumped.",
          doc2 = "This is the second document. ",
          doc3 = "This is a \"quoted\" text."
        )
spacy_parse(txt2, entity = TRUE, dependency = TRUE)

txt3 <- "We analyzed the Supreme Court with three natural language processing tools."
```

spacy_tokenize

Tokenize text with spaCy

Description

Efficient tokenization (without POS tagging, dependency parsing, lemmatization, or named entity recognition) of texts using spaCy.

Usage

spacy_tokenize(
  x,
  what = c("word", "sentence"),
  remove_punct = FALSE,
  remove_url = FALSE,
  remove_numbers = FALSE,
  remove_separators = TRUE,
  remove_symbols = FALSE,
  padding = FALSE,
  multithread = TRUE,
  output = c("list", "data.frame"),
  ...
)

Arguments

x a character object, a quanteda corpus, or a TIF-compliant corpus data.frame (see https://github.com/ropenscilabs/tif)
what the unit for splitting the text, available alternatives are:
  "word" word segmenter
  "sentence" sentence segmenter
remove_punct remove punctuation tokens.
remove_url remove tokens that look like a url or email address.
remove_numbers remove tokens that look like a number (e.g. "334", "3.1415", "fifty").
remove_separators remove spaces as separators when all other remove functionalities (e.g. remove_punct) have to be set to FALSE. When what = "sentence", this option will remove trailing spaces if TRUE.
remove_symbols remove symbols. The symbols are either SYM in pos field, or currency symbols.
spacy_uninstall

padding  if TRUE, leave an empty string where the removed tokens previously existed. This is useful if a positional match is needed between the pre- and post-selected tokens, for instance if a window of adjacency needs to be computed.

multithread  logical; If TRUE, the processing is parallelized using spaCy’s architecture (https://spacy.io/api)

output  type of returning object. Either list or data.frame.

...  not used directly

Value

either list or data.frame of tokens

Examples

```r
## Not run:
spacy_initialize()

# Tokenize a single text

txt <- "And now for something completely different."
spacy_tokenize(txt)

txt2 <- c(doc1 = "The fast cat catches mice.\nThe quick brown dog jumped.",
          doc2 = "This is the second document.",
          doc3 = "This is a "quoted" text."
        )
spacy_tokenize(txt2)

## End(Not run)
```

spacy_uninstall  Uninstall the spaCy environment

Description

Removes the virtual environment created by spacy_install()

Usage

`spacy_uninstall(confirm = interactive())`

Arguments

- `confirm`  logical; confirm before uninstalling spaCy?
Shorthand function to upgrade spaCy

Description

Upgrade spaCy (to a specific version).

Usage

```r
spacy_upgrade(
  version = "latest",
  lang_models = NULL,
  ask = interactive(),
  force = TRUE,
  ...
)
```

Arguments

- **version**: character; spaCy version to install (see details).
- **lang_models**: character; language models to be installed. Defaults `en_core_web_sm` (English model). A vector of multiple model names can be used (e.g., `c("en_core_web_sm", "de_core_news_sm")`). A list of available language models and their names is available from the spaCy language models page.
- **ask**: logical; ask whether to proceed during the installation. By default, questions are only asked in interactive sessions.
- **force**: ignore if spaCy/the lang_models is already present and install it anyway.
- **...**: passed on to `spacy_install()`.

Details

The function checks whether a suitable installation of Python is present on the system and installs one via `reticulate::install_python()` otherwise. It then creates a virtual environment with the necessary packages in the default location chosen by `reticulate::virtualenv_root()`.

If you want to install a different version of Python than the default, you should call `reticulate::install_python()` directly. If you want to create or use a different virtual environment, you can use, e.g., `Sys.setenv(SPACY_PYTHON = "path/to/directory")`.

See Also

`spacy_download_langmodel()`
Examples

```r
## Not run:
# install the latest version of spaCy
spacy_install()

# update spaCy
spacy_install(force = TRUE)

# install an older version
spacy_install(version = "3.1.0")

# install with GPU enabled
spacy_install(version = "cuda-autodetect")

# install on Apple ARM processors
spacy_install(version = "apple")

# install an old custom version
spacy_install(version = "[cuda-autodetect]==3.2.0")

# install several models with spaCy
spacy_install(lang_models = c("en_core_web_sm", "de_core_news_sm"))

# install spaCy to an existing virtual environment
Sys.setenv(RETICULATE_PYTHON = "path/to/python")
spacy_install()

## End(Not run)
```
Index

* datasets
  - data_char_paragraph, 3
  - data_char_sentences, 3

data_char_paragraph, 3
data_char_sentences, 3

entity_consolidate (entity_extract), 3
entity_extract, 3

nounphrase_consolidate
  (nounphrase_extract), 4
nounphrase_extract, 4

reticulate::install_python(). 11, 16
reticulate::virtualenv_root(). 11, 16

spacy_download_langmodel, 6
spacy_download_langmodel(), 11, 16
spacy_download_langmodel_virtualenv, 6
spacy_extract_entity, 7
spacy_extract_nounphrases, 8
spacy_finalize, 9
spacy_initialization, 10
spacy_install, 10
spacy_install(), 16
spacy_install_virtualenv, 12
spacy_parse, 12
spacy_parse(), 3–5
spacy_tokenize, 14
spacy_uninstall, 15
spacy_upgrade, 16
spacyr (spacyr-package), 2
spacyr-package, 2