Package ‘spacyr’

March 4, 2020

Type Package
Title Wrapper to the ‘spaCy’ 'NLP' Library
Version 1.2.1
Description An R wrapper to the 'Python' 'spaCy' 'NLP' library, from <http://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 7.0.2
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),
  European Research Council [fnd] (ERC-2011-StG 283794-QUANTESS)
Maintainer Kenneth Benoit <kbenoit@lse.ac.uk>
Repository CRAN
Date/Publication 2020-03-04 09:40:03 UTC
Description

An R wrapper to the Python (Cython) spaCy NLP system, from http://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
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.

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.

Description

Extract or consolidate entities from parsed documents

Usage

entity_extract(x, type = c("named", "extended", "all"), concatenator = "_")

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

Value

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 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

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")

# 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)

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

spacy_initialize()

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

# 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)

spacy_download_langmodel

Install a language model in a conda or virtual environment

Description

Installs one or more language models in a conda or virtualenv Python virtual environment as installed by spacy_install.
Usage

spacy_download_langmodel(
  model = "en",
  envname = "spacy_condaenv",
  conda = "auto"
)

spacy_download_langmodel_virtualenv(
  model = "en",
  envname = "spacy_virtualenv",
  virtualenv_root = NULL
)

Arguments

model name of the language model to be installed. A list of available language models and their names is available from the spaCy language models page.

envname name of the virtual environment

conda Path to conda executable. Default "auto" which automatically finds the path.

virtualenv_root path to the virtualenv environment to install spaCy language model. If NULL, the default path "~/.virtualenvs" will be 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/ropensci/tif)

output type of returned object, either "list" or "data.frame".
spacy_extract_nounphrases

<table>
<thead>
<tr>
<th>type</th>
<th>type of named entities, either named, extended, or all. See <a href="https://spacy.io/docs/usage/entity-recognition#entity-types">https://spacy.io/docs/usage/entity-recognition#entity-types</a> for details.</th>
</tr>
</thead>
<tbody>
<tr>
<td>multithread</td>
<td>logical; If TRUE, the processing is parallelized using spaCy’s architecture (<a href="https://spacy.io/api">https://spacy.io/api</a>)</td>
</tr>
</tbody>
</table>

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)

Value

either a list or data.frame of tokens

Examples

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")

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).
spacy_extract_nounphrases

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/ropensci/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

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

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")
**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**

```r
spacy_finalize()
```

**Author(s)**

Akitaka Matsuo

---

**spacy_initialize**

**Initialize spaCy**

**Description**

Initialize spaCy to call from R.

**Usage**

```r
spacy_initialize(
  model = "en_core_web_sm",
  python_executable = NULL,
  virtualenv = NULL,
  condaenv = NULL,
  ask = FALSE,
  refresh_settings = FALSE,
  save_profile = FALSE,
  check_env = TRUE,
  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`.
- `python_executable` the full path to the Python executable, for which spaCy is installed.
Install spaCy in conda or virtualenv environment

Description

Install spaCy in a self-contained environment, including specified language models. For macOS and Linux-based systems, this will also install Python itself via a "miniconda" environment, for `spacy_install`. Alternatively, an existing conda installation may be used, by specifying its path. The default setting of "auto" will locate and use an existing installation automatically, or download and install one if none exists.

For Windows, automatic installation of miniconda installation is not currently available, so the user will need to miniconda (or Anaconda) manually.

If you wish to install Python ion a "virtualenv", use the `spacy_install_virtualenv` function.

Usage

```r
spacy_install(
  conda = "auto",
  version = "latest",
  lang_models = "en_core_web_sm",
  python_version = "3.6",
  envname = "spacy_condaenv",
  pip = FALSE,
```
spacy_install

```r
python_path = NULL,
prompt = TRUE
)

spacy_install_virtualenv(
  version = "latest",
  lang_models = "en_core_web_sm",
  python_version = "3.6",
  python_path = NULL,
prompt = TRUE
)
```

Arguments

**conda** character; path to conda executable. Default "auto" which automatically find the path

**version** character; spaCy version to install. Specify "latest" to install the latest release, or "latest_v1" to install the latest release of spaCy v1.*. See spaCy Version Issues.

You can also provide a full major.minor.patch specification (e.g. "1.1.0")

**lang_models** character; language models to be installed. Default 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.

**python_version** character; determine Python version for condaenv installation. 3.5 and 3.6 are available.

**envname** character; name of the conda-environment to install spaCy. Default is "spacy_condaenv".

**pip** TRUE to use pip for installing spacy. If FALSE, conda package manager with conda-forge channel will be used for installing spacy.

**python_path** character; path to Python in virtualenv installation

**prompt** logical; ask whether to proceed during the installation

spaCy Version Issues

The version options currently default to the latest spaCy v2 (version = "latest"). As of 2018-04, however, some performance issues affect the speed of the spaCy pipeline for spaCy v2.x relative to v1.x. This can enormously affect the performance of spacy_parse(), especially when a large number of small texts are parsed. For this reason, the spacyr provides an option to automatically install the latest version of spaCy v1.*, using version = "latest_v1".

Examples

```r
# install spaCy in a miniconda environment (macOS and Linux)
spacy_install(lang_models = c("en_core_web_sm", "de_core_news_sm"), prompt = FALSE)

# install spaCy to an existing conda environment
```
spacy_install(conda = "~/anaconda/bin/")

## End(Not run)

## Not run:
# install spaCy in a virtualenv environment
spacy_install_virtualenv(lang_models = c("en_core_web_sm"))

## End(Not run)

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 tagsets (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

```r
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 `quanteda` corpus, or a TIF-compliant corpus data.frame (see https://github.com/ropensci/tif)
- **pos**: logical whether to return universal dependency POS tagset http://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).
spacy_tokenize

Tokenize text with spaCy

Description

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

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.

Value

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

Examples

spacy_initialize()

txt <- "And now for something completely different."
spacy_parse(txt)
spacy_parse(txt, pos = TRUE, tag = TRUE)
spacy_parse(txt, dependency = TRUE)

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_parse(txt3, entity = TRUE, nounphrase = TRUE)
spacy_parse(txt3, additional_attributes = c("like_num", "is_punct"))
spacy_tokenize

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/ropensci/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.
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
spacy_initialize()

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)
```

---

spacy_uninstall  

Uninstall spaCy conda environment

Description

Removes the conda environment created by `spacy_install()`

Usage

```r
spacy_uninstall(conda = "auto", prompt = TRUE, envname = "spacy_condaenv")
```

Arguments

- **conda**: path to conda executable, default to "auto" which automatically finds the path
- **prompt**: logical; ask whether to proceed during the installation
- **envname**: character; name of conda environment to remove

---

spacy_upgrade  

Upgrade spaCy in conda environment

Description

Upgrade spaCy in conda environment

Usage

```r
spacy_upgrade(
  conda = "auto",
  envname = "spacy_condaenv",
  prompt = TRUE,
  pip = FALSE,
  updateconda = FALSE,
  langmodels = "en_core_web_sm"
)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conda</td>
<td>Path to conda executable. Default &quot;auto&quot; which automatically find the path</td>
</tr>
<tr>
<td>envname</td>
<td>character; name of conda environment to upgrade spaCy</td>
</tr>
<tr>
<td>prompt</td>
<td>logical; ask whether to proceed during the installation</td>
</tr>
<tr>
<td>pip</td>
<td>TRUE to use pip for installing spacy. If FALSE, conda package manager with</td>
</tr>
<tr>
<td></td>
<td>conda-forge channel will be used for installing spacy.</td>
</tr>
<tr>
<td>update_conda</td>
<td>logical; If true, the conda binary for the system will be updated to the latest version. Default FALSE.</td>
</tr>
<tr>
<td>lang_models</td>
<td>Language models to be upgraded. Default NULL (No upgrade). A vector of multiple model names can be used (e.g. c(&quot;en_core_web_sm&quot;,&quot;de_core_web_sm&quot;))</td>
</tr>
</tbody>
</table>
Index

*Topic 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

spacy_download_langmodel, 5
spacy_download_langmodel_virtualenv
  (spacy_download_langmodel), 5
spacy_extract_entity, 6
spacy_extract_nounphrases, 7
spacy_finalize, 9
spacy_initialize, 9
spacy_install, 5, 10
spacy_install_virtualenv
  (spacy_install), 10
spacy_parse, 3–5, 12
spacy_tokenize, 13
spacy_uninstall, 15
spacy_upgrade, 15
spacyr (spacyr-package), 2
spacyr-package, 2