Package 'arete'

May 11, 2026

Title Automated REtrieval from TExt

Version 0.2

Date 2026-05-11

Author Vasco V. Branco [cre, aut] (ORCID:
<https://orcid.org/0000-0001-7797-3183>),
Vaughn Shirey [ctb] (ORCID: <https://orcid.org/0000-0002-3589-9699>),
Thomas Merrien [ctb] (ORCID: <https://orcid.org/0000-0002-0339-5656>),
Pedro Cardoso [aut] (ORCID: <https://orcid.org/0000-0001-8119-9960>)

Maintainer Vasco V. Branco <vasco.branco@helsinki.fi>

Description A Python based pipeline for extraction of species occurrence data through the usage of large language models. Includes validation tools designed to handle model hallucinations for a scientific, rigorous use of LLM. Currently supports usage of GPT with more planned, including local and non-proprietary models. For more details on the methodology used please consult the references listed under each function, such as Kent, A. et al. (1995) <doi:10.1002/asi.5090060209>, van Rijsbergen, C.J. (1979, ISBN:978-0408709293, Levenshtein, V.I. (1966) <https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf> and Klaus Krippendorff (2011) <https://repository.upenn.edu/handle/20.500.14332/2089>.

Depends R (>= 4.3.0)

Imports terra, cld2, stringr, reticulate, pdftools, fedmatch, kableExtra, dplyr, gecko, methods, ggplot2, jsonlite, googledrive, irr, rmarkdown

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Repository CRAN

Config/Needs/website rmarkdown

Suggests knitr

VignetteBuilder knitr

Date/Publication 2026-05-11 18:50:26 UTC

Contents

arete_data . . . 2
arete_package . . . 3
arete_setup . . . 5
aux_string_to_coords . . . 6
check_lang . . . 7
compare_IUCN . . . 8
create_training_data . . . 9
file_comparison . . . 10
gazetteer . . . 11
get_geodata . . . 12
install_OCR_packages . . . 13
install_python_packages . . . 14
labels . . . 15
labels_unique . . . 16
OCR_document . . . 16
performance_report . . . 17
process_document . . . 19
process_species_names . . . 20
string_to_coords . . . 21
WebAnnoTSV-class . . . 22
webanno_open . . . 22
webanno_summary . . . 23

Index **24**

`arete_data` *Example data packaged with arete*

Description

Load data included in the package. This includes:

1. `holzapfelae`, a txt file of a paper describing *Araneus holzapfelae* (Dippenaar-Schoeman & Foord, 2020).
2. `holzapfelae-extract`, a WebAnnoTSV file which is the output of processing `"holzapfelae"` with arete's LLM functionalities.
3. `annotations`, two files of annotated data of the same paper describing two species, *Sinlathrobium assingi* and *Sinlathrobium chenzhilini* (Chen *et al.* 2024).
4. `annotations-highlights` a version of the same two files reduced to just those sections containing annotated data.

Usage

```
arete_data(data = NULL)
```

Arguments

data	character. String of one of the data names mentioned in the description, e.g.: "holzapfelae". If NULL, the example files will be returned.

Value

Depending on data, either a character ("holzapfelae"), a WebAnnoTSV object ("holzapfelae-extract") or a list of WebAnnoTSV ("annotations", "annotations-highlights").

Source

This function is inspired by palmerpanguins::path_to_file() which in turn is based on readxl::readxl_example().

References

Dippenaar-Schoeman, A. S. & Foord, S. H. (2020a). First record of the orb-web spider Araneus holzapfelae Lessert, 1936 from South Africa (Arachnida: Araneidae). Sansa News 35: 19-21.

Chen, X., Ye, J.-P. and Peng, Z. (2024) 'Two new species and additional records of Sinlathrobium Assing (Coleoptera, Staphylinidae, Paederinae) from southern China', ZooKeys, 1218, pp. 25–33. doi:10.3897/zookeys.1218.128973.

Examples

```
arete_data()

arete_data("holzapfelae")
```

---

arete_package *Summary of methods in the arete package*

Description

Package arete seeks to provide easy Automated REtrieval of species data from TExt. To do this it processes user-supplied text, breaks it into API requests to LLM services and processes the output through a variety of machine learning and rule-based methods to deliever species data in a machine-readable format, ready for analysis. For a short and sweet use case of arete, try our vignette(package = "arete"). In broad terms, functions in arete can be placed in 6 different categories:

---

1. Setting up arete

arete_setup	Define a default virtual environment and install external dependencies
install_python_packages	Install or update python dependencies after setup
install_OCR_packages	Install or update the dependencies for our optional OCR utilities

2. Prepare annotation data

labels	Extract the labels and relations in a Webanno TSV 3.3 file to an easy, machine readable format rea
labels_unique	Extract all unique labels and relations in a Webanno TSV 3.3 file
webanno_open	Read the contents of a WebAnno TSV 3.3 file and create a webanno object, a format for annotated
webanno_summary	Summarize the contents of a group of WebAnno tsv files by counting labels and relations present
create_training_data	Open files with text and annotated data and build training data for large language models in a varie
file_comparison	Detect differences between two WebAnno files of the same text for annotation monitoring

3. Prepare text data

process_document	Extract text embedded in a .pdf or .txt file and proc
OCR_document	Optional utilities based on tesseract and nougatOCR
check_lang	Check if a given string is mostly (75% of the docume

4. Clean data

string_to_coords	Rule-based conversion of character strings containing geographic coordinates to sets of numeric
process_species_names	This function standardizes species names and fixes a number of common typos and mistakes that

5. Finetune and extract data

get_geodata	Call a Large Language Model (LLM) to extract species geographic data
gazetteer	Extract geographic coordinates from strings containing location names, using an online index

6. Evaluate model performance

performance_report	Produce a detailed report on the discrepancies between data extracted by a LLM and human annotate
compare_IUCN	Calculate EOO for two sets of coordinates for a practical assessment of data proximity

Contributors

The methods and functions in this package were written by Vasco Branco, with code contribuitions by Vaughn Shirey, Thomas Merrien. Code revision by Pedro Cardoso.

`arete_setup` *Setup arete*

Description

arete requires python to use most of its LLM utilities. OCR utilities will by default not be installed as they are not strictly necessary to the usage of the package. If interested please see install_OCR_packages(). We recommend that you install Python before running arete_setup. Usually performed through `sudo apt install python3-venv python3-pip python3-dev` on linux systems or `reticulate::install_python()` for other OS. Some external software might also be needed to successfully install arete's dependencies, including:

- GDAL, GEOS, PROJ, netcdf, sqlite3, tbb, see rspatial.github.io/terra
- poppler (rpm: poppler-cpp-devel (Fedora, CentOS, RHEL), brew: poppler (MacOS))
- kableExtra (fontconfig, deb:libfontconfig1-dev (Debian, Ubuntu, etc), rpm: fontconfig-devel (Fedora, EPEL), csw: fontconfig_dev (Solaris), brew: freetype (OSX))
- cmake

With some being covered by r-base-dev:

- gfortran, libgmp3-dev,
- harfbuzz freetype2 fribidi (deb:libharfbuzz-dev libfribidi-dev (Debian, Ubuntu, etc), rpm: harfbuzz-devel fribidi-devel (Fedora, EPEL), brew: harfbuzz fribidi (OSX))

Usage

```
arete_setup(python = NULL, last_tested = FALSE)
```

Arguments

python	character. Path to a python virtual environment in which to use arete.
last_tested	logical. If TRUE follows a requirements .txt file produced from the last working version.

Details

A custom virtual environment path may be passed to python but we recommend leaving it as NULL and using one of the paths found by reticulate. It is however useful if arete is already setup and you just wish to update its dependencies.

Value

character. Python path to the virtual environment created.

Examples

```
## Not run:
arete_setup()

## End(Not run)
```

---

aux_string_to_coords *Mechanical coordinate conversion*

Description

Mechanically convert character strings containing geographic coordinates and convert to sets of numeric values. Useful as most LLM experience better and more consistent results when asked to return a single string instead of two separate values.

Usage

```
aux_string_to_coords(coord_string, convert_from_dms = FALSE)
```

Arguments

coord_string	character. A string containg potential geographic coordinates.
convert_from_dms	logical. Boolean indicating whether strings should be converted from DD.mm.ss to DD.dddd format.

Details

Will convert all strings to N, W for consistency's sake. Future updates will probably make it a toggle.

Value

list. Contain latitude and longitude as the first and second elements, respectively.

Examples

```
example = "19 ° 34 ' S 29 ° 10 ° E"
aux_string_to_coords(example)
```

---

check_lang *Check if text is language-appropriate*

Description

Many, if not all, large language models are biased to English terms and sentence constructions. This function performs a quick check with `cld2` over every element of a string of characters and returns whether it is mostly (75

Usage

```
check_lang(strings, detailed = FALSE)
```

Arguments

`strings`	character. Vector of strings containing document sentences.
`detailed`	bool. If TRUE, the full cld2 report is returned as well.

Value

logical. If `TRUE` the language of the string is mostly English. If `detailed` is `TRUE` a list is instead returned for the full document.

Examples

```
# English
check_lang("Species Macrothele calpeiana is found in Alentejo.")

# Portuguese
check_lang("A espécie Macrothele calpeiana é encontrada no Alentejo.")
```

compare_IUCN *Check EOO differences between two sets of coordinates*

Description

Calculate EOO for two sets of coordinates for a practical assessment of data proximity.

Usage

```
compare_IUCN(x, y, plots = TRUE, verbose = TRUE)
```

Arguments

x	data.frame. With two columns containing latitude and longitude. Considered during reporting as data from a LLM.
y	data.frame. With the same formatting as x. Considered during reporting as the ground truth.
plots	logical. Determines if plots should be printed.
verbose	logical. Determines if output should be printed.

Details

Extent of occurrence (EOO) is defined as "the area contained within the shortest continuous imaginary boundary which can be drawn to encompass all the known, inferred or projected sites of present occurrence of a taxon, excluding cases of vagrancy"

Value

list with two values, the percentage of y that is part of the intersection with x and y and the percentage of x that is part of the intersection with x and y

Examples

```
set_a = matrix(
  c(54.30379, -25.48098, 54.21251,-25.47146, 59.53277, -20.37448, 55.59712,
 -22.39599, 55.47244, -26.30330, 61.39205,-21.12364, 56.24010, -24.40347),
 ncol = 2, byrow = TRUE
)

set_b = matrix(
c(54.30379,-25.48098, 111.42100,-19.00400, 54.21251, -25.47146, 59.53277,
  -20.37448, 55.59125, -22.39599, 55.47244,-26.30330, 61.39205, -21.12364,
   56.24010,-24.40347),
   ncol = 2, byrow = TRUE
 )

compare_IUCN(set_a, set_b)
```

create_training_data *Create training data for GPT*

Description

Open WebAnnoTSV files following RECODE structure and build training data for large language models in a variety of formats.

Usage

```
create_training_data(
  input,
  prompt = NULL,
  service = "GPT",
  aggregate = TRUE,
  export_type = "jsonl",
  out_path = NULL
)
```

Arguments

input	character or list. Either a set of paths to WebAnno TSV 3.3 files from which the text and annotated data are taken or a list with two terms, 1) paths to `.txt` or `.pdf` files e.g: `"./folder/file.pdf"` from which text data will be taken from and 2) paths to WebAnno TSV 3.3 files from which to take annotation data.
prompt	character. Custom prompt to be attached to each `text` during construction of the training data. Default prompt used otherwise.
service	character. Service to be used. Right now, only `GPT` is available.
aggregate	boolean. If TRUE and prompt is `"csv"`, a single csv is created.
export_type	character. Either `"jsonl"` or `"csv"`. If `"jsonl"`, a single file is created in which each line is a json specifying the input (`prompt` and `text`) and expected output (`data`).
out_path	character. Path to where the training data will be saved.

Value

matrix / data.frame

Examples

```
example = system.file(paste0("extdata/insecta_annot_1.tsv"), package = "arete")

create_training_data(input = example, service = "GPT", export_type = "jsonl")
```

`file_comparison` *Compare the contents of two WebAnno tsv files.*

Description

Detect differences between two WebAnno files of the same text for annotation monitoring.

Usage

```
file_comparison(
  input,
  method = "kripp",
  null_category = TRUE,
  log = NULL,
  verbose = TRUE
)
```

Arguments

`input`	list of character or WebAnnoTSV. The contents of WebAnno TSV v3.3 files as created by `webanno_open` or a set of paths leading to them.
`method`	character. A choice of "absolute" and "kripp".
`null_category`	logical. In cases where one annotator labels something and the remaining do not, should it be assigned a category or be set as `NA`?
`log`	character. Optional path to save the
`verbose`	boolean. Print the output of the function at the end.

Details

Right now, finds out the total sum of differences between all aspects of a given text. Method `kripp` calculates the Krippendorf Alpha, "a reliability coefficient developed to measure the agreement among observers, coders, judges, raters, or measuring instruments drawing distinctions among typically unstructured phenomena or assign computable values to them. alpha emerged in content analysis but is widely applicable wherever two or more methods of generating data are applied to the same set of objects, units of analysis, or items and the question is how much the resulting data can be trusted to represent something real" (Krippendorf, 2011).

Value

list. Later should be a dataframe with differences per block in the text.

References

Klaus Krippendorff (2011). Computing Krippendorff's Alpha-Reliability. Departmental Papers (ASC). University of Pennsylvania.

Examples

```
example = arete_data("annotations")

file_comparison(example)
```

---

gazetteer *Get geographic coordinates from localities*

Description

Extract geographic coordinates from strings containing location names, using an online index. through usage of a gazetteer for Python.

Usage

```
gazetteer(locality)
```

Arguments

locality	vector of characters. A vector of location names to process with a gazetteer.

Details

Gazetteers are geographical indexes, with the one used for `arete` being Nominatim which is accessible through the `GeoPy` client.

Value

matrix.

See Also

`arete_setup`

Examples

```
## Not run:
example = c("Lisbon", "London")

gazzetteer(example)
## End(Not run)
```

get_geodata *Call a Large Language Model (LLM) to extract species geographic data*

Description

Send an API request to extract species data from a document. For now only `service = "GPT"` is supported but more are planned including both proprietary and open source models.

Usage

```
get_geodata(
  path,
  user_key,
  service = "GPT",
  model = "gpt-3.5",
  tax = NULL,
  outpath = NULL,
  outliers = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	character. string of a file with species data in either pdf or txt format, e.g: `"./folder/file.pdf"`
`user_key`	list. Two elements, first element is a character with the user's API key, second element is a logical Bool determining whether the user's account has access to premium features. Both free keys and premium keys are allowed.
`service`	character. Model to be used. Right now, only requests using OpenAI's `chatGPT` are available.
`model`	character. Model name from given service to be used. You may use any of the models listed on OpenAI's developer platform. If you are unsure which model to use, we recommend picking `"gpt-3.5"` (default) or `"gpt-4o"`, as these will pick our recommended model from that version.
`tax`	character. Binomial name of the species to specify extraction to. Most often increases performance of the model.
`outpath`	Character string of a path to save output to in the format `"path/to/file/file_prefix"`.
`outliers`	logical. Whether or not results should be processed using the methods described in `gecko::outliers.detect()`
`verbose`	logical determining if output should be printed.

Details

The provided document is processed to remove invalid characters and multiple API requests may be made depending on the size. These API requests are independent from each other and this affects model performance. Ideally documents provided will fit entirely within the selected model's context window.

Value

matrix. Containing the extracted information.

See Also

arete_setup

Examples

```
## Not run:
file_path = arete_data("holzapfelae")

get_geodata(
  path = file_path,
  user_key = list(key = "your key here", premium = TRUE),
  model = "gpt-4o",
  outpath = "./out"
)
## End(Not run)
```

---

install_OCR_packages *Update OCR dependencies*

Description

Install python dependencies needed to run the optional OCR utilities in arete.

Usage

```
install_OCR_packages(envname = "arete", verbose = TRUE, ...)
```

Arguments

envname	character. Name or path to the Python virtual environment being used for arete.
verbose	logical. Determines if output should be printed.
...	Other parameters to pass to the installation function as per the documentation of py_install

Value

NULL. If any packages failed to install, a character listing which.

See Also

arete_setup

Examples

```
## Not run:
install_OCR_packages("./path/to/venv")

## End(Not run)
```

`install_python_packages` *Update python dependencies*

Description

Install python dependencies needed to run arete.

Usage

```
install_python_packages(envname, last_tested = FALSE, verbose = TRUE, ...)
```

Arguments

`envname`	character. Name or path to the Python virtual environment being used for arete.
`last_tested`	logical. If `TRUE` follows a requirements .txt file produced from the last working version.
`verbose`	logical. Determines if output should be printed.
`...`	Other parameters to pass to the installation function as per the documentation of `py_install`

Value

NULL. If any packages failed to install, a character listing which.

See Also

arete_setup

Examples

```
## Not run:
install_python_packages("./path/to/venv")

## End(Not run)
```

`labels` *Labels for model training*

Description

Extract the labels and relations in a webanno file to an easy, machine readable format ready for machine learning projects.

Usage

```
labels(
  data,
  label,
  relations = NULL,
  show_type = FALSE,
  show_tag = FALSE,
  show_ID = FALSE,
  handle_multiple = "duplicate"
)
```

Arguments

`data`	character or WebAnnoTSV. The contents of a WebAnno TSV v3.3 file as created by `webanno_open` or a path leading to it.
`label`	character. The main label. The relations must go FROM this term.
`relations`	character. The set of relations you'd like to extract.
`show_type`	logical. Add a column with the type of relation of the related terms.
`show_tag`	logical. Add a column with the tags of the related terms.
`show_ID`	logical. Add a column with the positional ID of the related terms.
`handle_multiple`	character. If there are multiple `relations` connecting to the same `label`, i.e. multiples locations, show should it be handled? Should `duplicate` rows be created or the content be `merge`'d?

Value

A list of dataframes, organized with columns for the corresponding line in the text, label and relations (if `relations != NULL`)

Examples

```
example = arete_data("annotations")[[1]]
labels(data = example, label = "Species", relations = "OCCURS")

labels(data = example,
label = c("TraitVal"), relations = c("meas_Sex"))
```

```
labels(data = example,
label = c("TraitVal"), relations = c("meas_trait", "meas_Sex", "meas_Unit"))

labels(data = example,
label = c("TraitVal"), relations = c("meas_trait", "meas_Sex", "meas_Unit"),
 handle_multiple = "merge")
```

`labels_unique` *Get the unique labels of a WebAnno document*

Description

Returns the unique classifications in the document

Usage

```
labels_unique(data, info = "all")
```

Arguments

`data`	character or WebAnnoTSV. The contents of a WebAnno TSV v3.3 file as created by `webanno_open` or a path leading to it.
`info`	character. A choice of "labels", "relations" and "all" on what to extract.

Value

list. With up to two elements, `$labels` and `$relations`, containing the respective elements.

Examples

```
example = arete_data("annotations")[[1]]
labels_unique(example, info = "all")
```

`OCR_document` *Scan PDF with optical character recognition (OCR)*

Description

Extract text contained under image form in a PDF through the use of optical character recognition software (OCR). Currently two options are available, `method = "nougat"` and `method = "tesseract"`.

Usage

```
OCR_document(in_path, out_path, method = "nougat", verbose = TRUE)
```

Arguments

in_path	character. string of a file with species data in either pdf or txt format, e.g: ./folder/file.pdf
out_path	character. Binomial name of the species used with applicable type.
method	character. Method used for the OCR. Currently it defaults to the only available method, nougatOCR.
verbose	logical. Print output after finish.

Details

For now OCR processing of documents is only supported on linux systems.

Value

character. Containing the extracted information.

See Also

arete_setup

Examples

```
## Not run:
OCR_document("path/to/file.pdf", "path/to/dir")

## End(Not run)
```

---

performance_report *Evaluate the performance of a LLM*

Description

Produce a detailed report on the discrepancies between LLM extracted data and human annotated data for the same collection of files.

Usage

```
performance_report(
  human_data,
  model_data,
  full_locations = "coordinates",
  string_distance = "levenshtein",
  verbose = TRUE,
  rmds = TRUE,
  path = NULL
)
```

Arguments

human_data	matrix. Ground truth dataset to compare the data extracted by a LLM.
model_data	matrix. Dataset of location data, following the description under human_data.
full_locations	character. Defines dataset structure. If "locations" then structure follows Species, Location, File. if "coordinates" then structure follows Species, Long, Lat, File. if "both" then structure follows Species, Location, Long, Lat, File.
string_distance	character. Selects the method through which the proximity between two strings is calculated, from those available under utils::adist().
verbose	logical. Determines if output should be printed.
rmds	logical. Determines if more extensive R Markdown files should be created at path.
path	character. Directory to which the output of the function is saved.

Details

Four main metrics are calculated to report on the performance of the model for coordinates. These are

- Accuracy, $\frac{TP}{TP+FP+FN}$, here defined as such in a system without True Negatives.
- Recall, $\frac{TP}{TP+FN}$, Kent *et al.* (1955)
- Precision, $\frac{TP}{TP+FP}$, Kent *et al.* (1955)
- F1 score, $\frac{2}{\frac{1}{Precision}+\frac{1}{Sensitivity}}$, van Rijsbergen(1979).

Additional metrics are calculated, including: 1) a distance-weighed confusion matrix where the sum of each type of error (False Negatives and False Positives) is done by weights, calculated to be inverse to the mean euclidean distance of that data point to all others. This way errors that are close to existing data for that species will count less than those further way, i.e. a data point was hallucinated that was close to existing data or, a data point was missed that is already represented in the data. This adjusted confusion matrix is also presented along with versions of the four main metrics calculated with these values. To report on the performance of locations, by default the minimum Levenshtein distance (Levenshtein, 1966) between a term and all other terms is calculated. Which is defined as:

$$lev(a,b) = \begin{cases} |a| & if|b| = 0, \\ |b| & if|a| = 0, \\ lev(tail(a), tail(b)) & if head(a) = head(b), \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & otherwise \end{cases}$$

In short, the number of edits needed to turn one string a into string b.

Value

list. A confusion matrix is returned for every species per document, plus one for the entire process.

References

- Kent, A. et al. (1955). "Machine literature searching VIII. Operational criteria for designing information retrieval systems", *American Documentation*, 6(2), pp. 93–101. doi:10.1002/asi.5090060209.
- van Rijsbergen, C.J. (1979). "Information Retrieval", Architectural Press. ISBN: 978-0408709293.
- Levenshtein, V.I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Physics-Doklady*, 10(8), pp. 707–710 [Translated from Russian].

Examples

```
trial_data = arete::arete_data("holzapfelae-extract")
trial_data = cbind(trial_data[,1:2], arete::string_to_coords(trial_data[,3])[2:1], trial_data[,4:5])

trial_data = list(
  GT = trial_data[trial_data$Type == "Ground truth", 1:5],
  MD = trial_data[trial_data$Type == "Model", 1:5]
)

# make sure you run arete_setup() beforehand!
performance_report(
  trial_data$GT,
  trial_data$MD,
  full_locations = "both",
  verbose = FALSE,
  rmds = FALSE
)
```

---

process_document *Extract and process text from a document*

Description

This function extracts text embedded in a `.pdf` or `.txt` file and processes it so it can be safely used by LLM API's.

Usage

```
process_document(path, extra_measures = NULL)
```

Arguments

`path`	character. Path leading to the desired PDF file.
`extra_measures`	character. To be implemented. Some documents are especially difficult for LLM to process due to a variety of issues such as size and formatting. `extra_measures` tries to improve future performance by cropping the document given to only the central passage mentioning a specific species. `"header"` and, by extension, `"both"` require an mmd file that is the output of nougatOCR.

Value

character. Fully processed text.

Examples

```
path = arete_data("holzapfelae")
process_document(path)

extra_measures = list("mention", "Tricholathys spiralis")
```

`process_species_names` *Process and fix species names*

Description

This function standardizes species names and fixes a number of common typos and mistakes that commonly occur due to OCR.

Usage

```
process_species_names(species, processing = "all", enforce = FALSE)
```

Arguments

`species`	character. String with the name of the species to be processed.
`processing`	character. One of `"trim"`, `"sp nov"`, `"hyphen broken"`, `"abbreviate"`, `"spacing"` and `"all"`. By default set to `"all"`, performing every edit.
`enforce`	boolean. Remove non-conforming entries before returning.

Value

character. A string with the standardized species name.

Examples

```
process_species_names("macro - thele calpeiana sp. nov.")
```

`string_to_coords` *Convert strings to numerical coordinates*

Description

Convert character strings containing geographic coordinates to sets of numeric values through a rule-based method. Useful as most LLM experience better and more consistent results when asked to return a single string instead of two separate values.

Usage

```
string_to_coords(
  coord_vector,
  verbose = TRUE,
  convert_from_dms = FALSE,
  log = FALSE
)
```

Arguments

`coord_vector`	character. A string containing potential geographic coordinates.
`verbose`	logical. Whether or not to print the overall success of the function upon end.
`convert_from_dms`	logical. If TRUE the function will also attempt to convert coordinates from decimal minutes seconds decimal degrees. A small accuracy error will apply otherwise.
`log`	logical. If TRUE function will return a vector of failed conversions as well.

Details

Will convert all strings to N, W for consistency's sake. In a future update the user will be able to specify the outcome format, e.g: "39°S 42°E".

Value

list. Contain latitude and longitude as the first and second elements, respectively.

Examples

```
example = c('N 39degrees05'53" E 26degrees21'57"', "39 ° 80 ' N , 32 ° 78 ' E",
"30 ° 20'54.32 \" N , 78 ° 0'53.37 \" E", "19 ° 34 ' S 29 ° 10 ° E",
"- 25.05012 S ; - 65.49622 W", "S 12 ° 06 ' , W 76 ° 57 '",
"19 ° 34 ' S 29 ° 10 ° E", "19 ° 32 ' S 29 ° 08 ° E")
string_to_coords(example)
```

WebAnnoTSV-class *WebAnno TSV v3.3 class creator.*

Description

Crop raster layers to minimum size possible and uniformize NA values across layers.

webanno_open *Open a WebAnno TSV v3.3 file.*

Description

Read the contents of a WebAnno TSV 3.3 file, a format for annotated text containing named entities and relations.

Usage

```
webanno_open(path, cut_to_content = FALSE)
```

Arguments

path	character. A path to a WebAnno TSV v3.3 file.
cut_to_content	logical. Restrict the output file to those sentences containing annotated labels and relations.

Details

One of the format types under use at https://inception-project.github.io/.

Value

WebAnnoTSV. A list of dataframes, each named after the corresponding paragraph in the text.

Examples

```
example = system.file(paste0("extdata/insecta_annot_1.tsv"), package = "arete")

webanno_open(example)
```

webanno_summary *Summarize the contents of a group of WebAnno tsv files*

Description

Returns

Usage

```
webanno_summary(input, goal = "relations", collected = TRUE, verbose = TRUE)
```

Arguments

input	list of character or WebAnnoTSV. The contents of WebAnno TSV v3.3 files as created by webanno_open or a set of paths leading to them.
goal	character. One of "labels", "relations" or "all" determining what is summarized.
collected	boolean. Organize the summary under a single dataframe, otherwise a list is returned.
verbose	boolean. Print the final output.

Value

dataframe.

Examples

```
example = arete_data("annotations")

webanno_summary(example)
```

Index

∗ **large language models**
    arete_package, 3
∗ **package**
    arete_package, 3

arete *(*arete_package*)*, 3
arete_data, 2
arete_package, 3
arete_setup, *4*, 5, *11*, *13, 14*, *17*
aux_string_to_coords, 6

check_lang, *4*, 7
cld2, *7*
compare_IUCN, *5*, 8
create_training_data, *4*, 9

file_comparison, *4*, 10

gazetteer, *5*, 11
gecko::outliers.detect(), *12*
get_geodata, *5*, 12

install_OCR_packages, *4*, 13
install_python_packages, *4*, 14

labels, *4*, 15
labels_unique, *4*, 16

OCR_document, *4*, 16

palmerpanguins::path_to_file(), *3*
performance_report, *5*, 17
process_document, *4*, 19
process_species_names, *4*, 20
py_install, *13, 14*

readxl::readxl_example(), *3*

string_to_coords, *4*, 21

utils::adist(), *18*

webanno_creator *(*WebAnnoTSV-class*)*, 22
webanno_open, *4*, *10*, *15, 16*, 22, *23*
webanno_summary, *4*, 23
WebAnnoTSV-class, 22