Package ‘interlineaR’

October 13, 2022

Type Package

Title Importing Interlinearized Corpora and Dictionaries as Produced by Descriptive Linguistics Software

Version 1.0

Date 2018-05-18

Description Interlinearized glossed texts (IGT) are used in descriptive linguistics for representing a morphological analysis of a text through a morpheme-by-morpheme gloss. 'InterlineaR' provide a set of functions that targets several popular formats of IGT ('SIL Toolbox', 'EMELD XML') and that turns an IGT into a set of data frames following a relational model (the tables represent the different linguistic units: texts, sentences, word, morphems).
The same pieces of software ('SIL FLEX', 'SIL Toolbox') typically produce dictionaries of the morphemes used in the glosses. 'InterlineaR' provide a function for turning the LIFT XML dictionary format into a set of data frames following a relational model in order to represent the dictionary entries, the sense(s) attached to the entries, the example(s) attached to senses, etc.

License BSD_3_clause + file LICENSE

Depends R (>= 2.14), xml2, reshape2

LazyData true

URL https://github.com/sylvainloiseau/interlineaR

Collate read.emeld.R read.lift.R interlineaR-package.R read.toolbox.R read.pangloss.R dataset.R lift.specifications.R common.R

BuildResaveData xz

Encoding UTF-8

RoxygenNote 6.0.1

Suggests kableExtra, knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Sylvain Loiseau [aut, cre]

Maintainer Sylvain Loiseau <sylvain.loiseau@univ-paris13.fr>

Repository CRAN

Date/Publication 2018-05-22 12:17:37 UTC

R topics documented:

`interlineaR` *Importing interlinearized corpora and dictionaries as produced by descriptive linguistics software*

Description

Importing interlinearized corpora and dictionaries as produced by descriptive linguistics software

Author(s)

Maintainer: Sylvain Loiseau <`sylvain.loiseau@univ-paris13.fr`>

See Also

Useful links:

- `https://github.com/sylvainloiseau/interlineaR`

`lift-format` *Information about the structure of the LIFT XML format in order to easily generate XPath expression and extract information.*

Description

There are four fonctions: one for each table to be built (entries, senses, examples, relations).

Usage

```
entry.fields.spec()

sense.fields.spec()

example.fields.spec()

relation.fields.spec()
```

Details

Each functions return a table with the following columns: A name for this field "Path": an XPath expression toward an element. "Type" how to retreive the content of this element in some frequent cases: - "form" indicates that the content is in ./form/text; form contains an attribute @lang with either vernacular languages code(s), or analysis language code(s). In this case, the Sub-type column state vernacular of analysis accordingly. - "trait" indicate that the content is in a @value attribute; the trait has a "name" attribute give in the Sub-type column. - "gloss" is similar to "form" above. "Sub-type": in the cases where Type has the values "form" or "gloss", indicats if @lang is vernacular ou analysis; in the cases where Type has the value "trait" : the value of @name. "Concat" an XPath expression for building the value with the element using XPath concat() "Collapse": TRUE = element may appears several time and have to be collapsed in order to build the cell value.

Value

a data.frame

a data.frame

a data.frame

a data.frame

`lift.specification` *List of the available pieces of information for each entry (ie column in the entry table)*

Description

List of the available pieces of information for each entry (ie column in the entry table)

List of the available pieces of information for each sense (ie column in the sense table)

List of the available pieces of information for each example (ie column in the example table)

List of the available pieces of information for each relation (ie column in the relation table)

Usage

```
available.entry.fields()

available.sense.fields()

available.example.fields()

available.relation.fields()
```

Value

a character vector of entries.

a character vector of entries.

a character vector of entries.

a character vector of entries.

read.emeld *Read an EMELD XML document containing an interlinearized corpus.*

Description

The EMELD XML vocabulary has been proposed for the encoding of interlinear glosses. It is used by the FieldWorks software (SIL FLEX) as an export format.

Usage

```
read.emeld(file, vernacular.languages, analysis.languages = "en",
  get.morphemes = TRUE, get.words = TRUE, get.sentences = TRUE,
  get.texts = TRUE, text.fields = c("title", "title-abbreviation", "source",
  "comment"), sentence.fields = c("segnum", "gls", "lit", "note"),
  words.vernacular.fields = "txt", words.analysis.fields = c("gls", "pos"),
  morphemes.vernacular.fields = c("txt", "cf"),
  morphemes.analysis.fields = c("gls", "msa", "hn"), sep = ";")
```

Arguments

`file` the path (or url) to a document in ELMED vocabulary

`vernacular.languages` character vector: one or more codes of languages analysed in the document.

`analysis.languages` character vector: one or more codes of languages used for the analyses (in glosses, translations, notes) in the document.

`get.morphemes` logical vector: should the returned list include a slot for the description of morphemes?

`get.words` logical vector: should the returned list include a slot for the description of words?

`get.sentences` logical vector: should the returned list include a slot for the description of sentences?

`get.texts` logical vector: should the returned list include a slot for the description of texts?

`text.fields` character vector: information to be extracted for the texts (and turned into corresponding column in the data.frame describing texts) The default are:

- "title"
- "title-abbreviation"
- "source"
- "comment"

`sentence.fields` character vector: information to be extracted for the sentences (and turned into corresponding column in the data.frame describing sentences) The default are:

- "segnum" : an ID of the sentende

- "gls": a translation (possibly in all analysis languages)
- "lit": a litteral translation (possibly in all analysis languages)
- "note": note (possibly in all analysis languages)

`words.vernacular.fields`
character vector: information (in vernacular language(s)) to be extracted for the words (and turned into corresponding columns in the data.frame describing words) The default are:

- "txt" : the original text

`words.analysis.fields`
character vector: information (in analysis language(s)) to be extracted for the words (and turned into corresponding columns in the data.frame describing words) The default are:

- "gls" : a gloss of the word
- "pos" : the part of speech of the word

`morphemes.vernacular.fields`
character vector: information (in vernacular language(s)) to be extracted for the morphemes (and turned into corresponding columns in the data.frame describing morphemes). May be null or empty.

- "txt" : the text of the morpheme
- "cf" : the canonical form of the morpheme

`morphemes.analysis.fields`
character vector: information (in analysis language(s)) to be extracted for the morphemes (and turned into corresponding columns in the data.frame describing morphemes). May be null or empty.

- "gls" : the gloss of the morpheme
- "msa" : the part of speech of the morpheme
- "hn" : a number for the identifiation of the morpheme amongst its homophone.

`sep` character vector: the character used to join multiple notes in the same language.

Details

If several 'note' fields in the same language are present in a sentence, they will be concatenated (see the "sep" argument)

Value

a list with slots named "morphemes", "words", "sentences", "texts" (some slot may have been excluded throuth the "get.*" arguments, see above). Each slot is a data.frame containing the information on the corresponding unit. In each data.frame, each row describe an occurrence (the first row of the result$morphemes data.frame describe the first morpheme of the corpus). In each data.frame, the first columns give ids refering to the line in other data.frame (so that we can link the first morpheme to the text, the sentence or the word it belongs to). The following columns give information about the corresponding occurrence of the unit. Which information are extracted from the document and included in the data frame depends upton the *.fields parameters (see above). Columns made are coined using the field name and the language code. For instance, if read.emeld is called with the parameters vernacular.languages="tww" and morphemes.vernacular.fields=c("txt", "cf"), then the column txt.tww and cf.tww will be created in the morphemes slot data frame.

References

Baden Hughes, Steven Bird and Catherine Bow *Encoding and Presenting Interlinear Text Using XML Technologies*, http://www.aclweb.org/anthology/U03-1008

SIL FieldWorks: https://software.sil.org/fieldworks/

Examples

```
path <- system.file("exampleData", "tuwariInterlinear.xml", package="interlineaR")
corpus <- read.emeld(path, vernacular="tww", analysis="en")
head(corpus$morphemes)

# In some cases, one may have to combine information coming from various data.frame.
# Lets imagine one needs to have in the same data.frame the morphemes data
# plus the "note" field attached to sentences:
# - The easy way is to combine all the columns of the two data frame 'morphemes' and 'sentence' :
combined <- merge(corpus$morphemes, corpus$sentences, by.x="sentence_id", by.y="sentence_id")
head(combined)

# - Alternatively, one may use vector extraction in order to add only the desired column
# to the morphemes data frame:
corpus$morphemes$note = corpus$sentences$note.en[ corpus$morphemes$sentence_id ]
head(corpus$morphemes)
```

read.lift *Parse a dictionary in XML LIFT (Lexicon Interchange FormaT) vocabulary and turn it into a set of data.frame*

Description

The dictionary is turned into a list of up to four data frame: "entries", "senses", "examples" and "relations". The data frame are pointing to each other through IDs, following a relational data model.

Usage

```
read.lift(file, vernacular.languages, analysis.languages = "en",
  get.entry = TRUE, get.sense = TRUE, get.example = TRUE,
  get.relation = TRUE, entry.fields = available.entry.fields(),
  sense.fields = available.sense.fields(),
  example.fields = available.example.fields(),
  relation.fields = available.relation.fields(), simplify = FALSE,
  sep = ";")
```

Arguments

`file` : a length-one character vector containing the path to a LIFT XML document.

`vernacular.languages` character vector: the code of the vernacular language.

`analysis.languages` character vector: code of the object language used in the glosses and analyses.

`get.entry` logical length-1 vector: include the entries table in the result?

`get.sense` logical length-1 vector: include the senses table in the result?

`get.example` logical length-1 vector: include the examples table in the result?

`get.relation` logical length-1 vector: include the relations table in the result?

`entry.fields` character vector: names of the fields to be included in the entries table. See available.entry.fields() for the complete list of the available fields.

`sense.fields` character vector: names of the fields to be included in the senses table. See available.sense.fields() for the complete list of the available fields.

`example.fields` character vector: names of the fields to be included in the examples table. See available.example.fields() for the complete list of the available fields.

`relation.fields` character vector: names of the fields to be included in the relations table. See available.relation.fields() for the complete list of the available fields.

`simplify` logical length-1 vector: if true, columns containing only empty values are removed from all data frame.

`sep` character vector: the character used to join multiple notes in the same language.

Details

"Field" in this document denote a piece of information in LIFT, such as the "gloss" in a sense or "citation form" of an entry. A field may correspond to several columns in the resulting data frame, since fields are multilingual. "gloss" is an analysis field, thus if two analysis.languages are declared, for instance "en" and "fr", then two columns will be present, gloss.en and gloss.fr, in the senses data frame. The "citation form" field, on the other hand, is an vernacular language field, thus if several vernacular fields are declared, several form columns will be present in the entries data frame.

Value

a list with up to four slots named "entries", "senses", "examples" and "relations", each slot containing a data.frame

References

http://code.google.com/p/lift-standard

See Also

write.CLDF for serialization

Examples

```
path <- system.file("exampleData", "tuwariDictionary.lift", package="interlineaR")
dictionary <- read.lift(path, vernacular.languages="tww")

# Reduce the size of the data frames by filtering to columns actually containing something...
dictionary <- read.lift(path, vernacular.languages="tww", simplify=TRUE)

# Get information in the different analysis languages used in the document (english and tok pisin)
dictionary <- read.lift(path, vernacular.languages="tww", analysis.languages=c("en", "tpi"))

# Restrict to entries and senses dataframe, and explicitly ask for some fields:
dictionary <- read.lift(
  path,
  vernacular.languages="tww",
  get.example=FALSE,
  get.relation=FALSE,
  entry.fields=c("lexical-unit", "morph-type"),
  sense.fields=c("grammatical-info.value", "gloss", "definition",
  "semantic-domain-ddp4", "grammatical-info.traits")
)
```

---

read.pangloss *Read a file in the format used in the pangloss collection*

Description

The pangloss collection (http://lacito.vjf.cnrs.fr/pangloss/index_en.html) is a large collection of interlinearized texts.

Usage

```
read.pangloss(url, DOI = NULL, get.texts = TRUE, get.sentences = TRUE,
  get.words = TRUE, get.morphemes = TRUE)
```

Arguments

`url`	a length one character vector with the url of the document to be imported
`DOI`	an unique identifier
`get.texts`	should the ’texts’ data.frame be included in the result ?
`get.sentences`	should the ’sentences’ data.frame be included in the result ?
`get.words`	should the ’words’ data.frame be included in the result ?
`get.morphemes`	should the ’morphemes’ data.frame be included in the result ?

Value

a list with up to 5 slots corresponding to different units and named "texts", "sentences", "words", "morphemes". Each slot contains a data frame where each line describe an occurrence of the corresponding unit.

References

http://lacito.vjf.cnrs.fr/pangloss/index_en.html

Examples

```
path <- system.file("exampleData", "FOURMI.xml", package="interlineaR")
corpus <- read.pangloss(path)
head(corpus$morphemes)
```

---

read.toolbox *Parse a Toolbox (SIL) text file*

Description

Parse a Toolbox (SIL) text file

Usage

```
read.toolbox(path, text.fields.suppl = NULL, sentence.fields.suppl = c("tx",
  "nt", "ft"), word.fields.suppl = NULL, morpheme.fields.suppl = NULL)
```

Arguments

`path` length-1 character vector: the path to a toolbox text file.

`text.fields.suppl`
character vector: the code of supplementary fields to be searched for each text (genre, ...). "id" is mandatory and need not to be listed here.

`sentence.fields.suppl`
character vector: the code of supplementary fields to be searched for each sentence (such as ft, nt). "ref" is mandatory and need not to be listed here.

`word.fields.suppl`
character vector: the code of supplementary fields to be searched for each word. "tx" is mandatory and need not to be listed here.

`morpheme.fields.suppl`
character vector: the code of supplementary fields to be searched for each morpheme. "mb", "ge", "ps" are mandatory and need not to be listed here.

Value

a list with four slots "texts", "sentences", "words" and "morphemes", each one containing a data frame. In these data frame, each row describe an occurrence of the corresponding unit.

References

https://software.sil.org/toolbox/

See Also

read.emeld (XML vocabulary for interlinearized glossed texts)

Examples

```
corpuspath <- system.file("exampleData", "tuwariToolbox.txt", package="interlineaR")
corpus <- read.toolbox(corpuspath)
```

vatlongos *A corpus of 10 texts of the Vatlongos (vtk) language*

Description

The corpus is produced with the read.emeld() function. It is a list of 4 slots representing four units: "texts" "sentences" "words" "morphems". Each slot contains a data frame, and each row in the data.frame describe one occurrences of the corresponding unit.

Usage

```
vatlongos
```

Format

A list with 4 slots

Details

- texts : a data frame of 95 units and 5 columns ("text_id", "title.en", "title.abbreviation.en", "source.en", "comment.en")
- sentenes : a data frame of 3967 units and 6 columns ("text_id", "sentence_id", "segnum.en", "gls.en", "lit.en", "note.en")
- words : a data frame of 52983 units and 6 columns ("text_id" "sentence_id" "word_id" "txt.tvk" "gls.en" "pos.en")
- mophems numeric : a data frame of 56354 units and 10 columns ("text_id" "sentence_id" "word_id" "morphem_id" "type" "txt.tvk" "cf.tvk" "gls.en" "msa.en" "hn.en")

See the vignette vatlongos for Case study based on this corpus.

References

Eleanor Ridge <Eleanor_Ridge@soas.ac.uk>

Index