

Package ‘labourR’

October 13, 2022

Type Package

Title Classify Multilingual Labour Market Free-Text to Standardized Hierarchical Occupations

Version 1.0.0

Description Allows the user to map multilingual free-text of occupations to a broad range of standardized classifications. The package facilitates automatic occupation coding (see, e.g., Gweon et al. (2017) <[doi:10.1515/jos-2017-0006](https://doi.org/10.1515/jos-2017-0006)> and Turrell et al. (2019) <[doi:10.3386/w25837](https://doi.org/10.3386/w25837)>), where the ISCO to ESCO mapping is exploited to extend the occupations hierarchy, Le Vrang et al. (2014) <[doi:10.1109/mc.2014.283](https://doi.org/10.1109/mc.2014.283)>. Document vectorization is performed using the multilingual ESCO corpus. A method based on the nearest neighbor search is used to suggest the closest ISCO occupation.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

URL <https://github.com/AleKoure/labourR>

BugReports <https://github.com/AleKoure/labourR/issues>

Suggests knitr, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder knitr

Depends R (>= 3.1.0)

Imports data.table, cld2, magrittr, stopwords, stringdist

NeedsCompilation no

Author Alexandros Kouretsis [aut, cre],
Andreas Bampouris [aut],
Petros Morfiris [aut],
Konstantinos Papageorgiou [aut],
Stavros Ladas [ctb],
Athanasios Siaperas [ctb],
Philippe Tissot [ctb],
Nikos Vaslamatzis [ctb],
Eworx S.A [cph]

Maintainer Alexandros Kouretsis <ako@eworx.gr>

Repository CRAN

Date/Publication 2020-07-18 09:30:07 UTC

R topics documented:

classify_occupation	2
cleansing_corpus	4
get_language_code	5
get_stopwords	5
identify_language	6
isco_occupations_bundle	6
occupations_bundle	7
tf_idf	8

Index **10**

classify_occupation *Classify occupations*

Description

This function takes advantage of the hierarchical structure of the ESCO-ISCO mapping and matches multilingual free-text with the **ESCO** occupations vocabulary in order to map semi-structured vacancy data into the official ESCO-ISCO classification.

Usage

```
classify_occupation(
  corpus,
  id_col = "id",
  text_col = "text",
  lang = "en",
  num_leaves = 10,
  isco_level = 3,
  max_dist = 0.1,
  string_dist = NULL
)
```

Arguments

corpus	A data.frame or a data.table that contains the id and the text variables.
id_col	The name of the id variable.
text_col	The name of the text variable.
lang	The language that the text is in.

num_leaves	The number of occupations/neighbors that are kept when matching.
isco_level	The ISCO level of the suggested occupations. Can be either 1, 2, 3, 4 for ISCO occupations, or NULL that returns ESCO occupations.
max_dist	String distance used for fuzzy matching. The <code>amatch</code> function from the <code>stringdist</code> package is used.
string_dist	String dissimilarity measurement. Available string distance metrics: stringdist-metrics .

Details

First, the input text is cleansed and tokenized. The tokens are then matched with the ESCO occupations vocabulary, created from the preferred and alternative labels of the occupations. They are joined with the `tfidf` weighted tokens of the ESCO occupations and the sum of the tf-idf score is used to retrieve the suggested ontologies. Technically speaking, the suggested ESCO occupations are retrieved by solving the optimization problem,

$$\arg \max_d \{ \vec{u}_{binary} \cdot \vec{u}_d \}$$

where, \vec{u}_{binary} stands for the binary representation of a query to the ESCO-vocabulary space, while, \vec{u}_d is the ESCO occupation normalized vector generated by the tf-idf numerical statistic. If an ISCO level is specified, the k-nearest neighbors algorithm is used to determine the suggested occupation, classified by a plurality vote in the corresponding hierarchical level of its neighbors.

Before the suggestions are returned, the preferred label of each suggested occupation is added to the result, using the `occupations_bundle` and `isco_occupations_bundle` as look-up tables.

Value

Either a `data.table` with the id, the preferred label and the suggested ESCO occupation URIs (num_leaves predictions for each id), or a `data.table` with the id, the preferred label and the suggested ISCO group of the inputted level (one for each id).

References

M.P.J. van der Loo (2014). [The stringdist package for approximate string matching](#). R Journal 6(1) pp 111-122.

Gweon, H., Schonlau, M., Kaczmirek, L., Blohm, M., & Steiner, S. (2017). [Three Methods for Occupation Coding Based on Statistical Learning](#), *Journal of Official Statistics*, 33(1), 101-122.

Arthur Turrell, Bradley J. Speigner, Jyldyz Djumalieva, David Copple, James Thurgood (2019). [Transforming Naturally Occurring Text Data Into Economic Statistics: The Case of Online Job Vacancy Postings](#).

ESCO Service Platform - [The ESCO Data Model documentation](#)

Examples

```
corpus <- data.frame(
  id = 1:3,
  text = c(
    "Junior Architect Engineer",
    "Cashier at McDonald's",
```

```
    "Priest at St. Martin Catholic Church"
  )
)
classify_occupation(corpus = corpus, isco_level = 3, lang = "en", num_leaves = 5)
```

cleansing_corpus *Cleansing Corpus*

Description

The function performs text cleansing by removing escape characters, non alphanumeric, longwords, excess space, and turns all letters to lower case.

Usage

```
cleansing_corpus(
  text,
  escape_chars = TRUE,
  nonalphanum = TRUE,
  longwords = TRUE,
  whitespace = TRUE,
  tolower = TRUE
)
```

Arguments

text	Character vector of free text to be cleansed.
escape_chars	If TRUE, removes escape characters for slash n, slash r and slash t.
nonalphanum	If TRUE, removes non-alphanumeric characters.
longwords	If TRUE, removes words with more than 35 characters.
whitespace	If TRUE, removes excess whitespace.
tolower	If TRUE, turns letters to lower.

Value

A character vector of the cleansed text.

Examples

```
txt <- "It has roots in a piece of classical Latin literature from 45 BC"
cleansing_corpus(txt)
```

get_language_code	<i>Get language code from file name</i>
-------------------	---

Description

Occupations' labels and structure are exposed at the ESCO web portal. This function retrieves languages from the downloadable CSVs, see [escopedia](#).

Usage

```
get_language_code(string)
```

Arguments

string	Filepath with a language code as given by ESCO downloadable .CSVs.
--------	--

Value

A character vector with two-letter language codes.

Examples

```
get_language_code("occupations_en.csv")
```

get_stopwords	<i>Retrieve stopwords</i>
---------------	---------------------------

Description

The functions retrieves stopwords from the [stopwords](#) package using the ISO-639-1 encoding. For miscellaneous languages [data_stopwords_misc](#) are used.

Usage

```
get_stopwords(code)
```

Arguments

code	A string with the language code of the stopwords.
------	---

Value

Character vector with the stopwords or NULL if the language code is unknown.

Examples

```
get_stopwords("en")[1:10]
```

identify_language *Detect Language*

Description

This function performs language detection by using Compact Language Detector 2 from CRAN library [cld2](#). It is vectorised and guesses the language of each string. Note that it is not designed to do well on very short text, lists of proper names, part numbers, etc. CLD2 has the highest F1 score and is an order of magnitude faster than CLD3.

Usage

```
identify_language(text)
```

Arguments

`text` A string with text to classify or a connection to read from.

- `cld2`: Probabilistically (Naïve Bayesian classifier) detects over 80 languages in plain text.

Value

A character vector with ISO-639-1 two-letter language codes.

Examples

```
txt <- c("English is a West Germanic language ", "In espaniol, le lingua castilian")
identify_language(txt)
```

isco_occupations_bundle
ISCO occupations bundle

Description

The International Standard Classification of Occupations (ISCO) is a four-level classification of occupation groups managed by the International Labour Organisation (ILO). Its structure follows a grouping by education level. The two latest versions of ISCO are ISCO-88 (dating from 1988) and ISCO-08 (dating from 2008).

- The ESCO version used is ESCO v1 1.0.3 retrieved at 11/12/2019.

Usage

```
isco_occupations_bundle
```

Format

A data.table with 2 variables, which are:

iscoGroup Four-level classification of occupation groups.

preferredLabel Preferred name of ISCO occupation concepts.

Source

International Standard Classification of Occupations (**ISCO**).

occupations_bundle *ESCO occupations bundle*

Description

The occupations pillar is one of the three pillars of **ESCO**. It organizes the occupation concepts. It uses hierarchical relationships between them, metadata as well as mappings to the International Standard Classification of Occupations **ISCO** in order to structure the occupations. The descriptions of each concept is provided only in English.

- The ESCO version used is ESCO v1 1.0.3 retrieved at 11/12/2019.

Usage

occupations_bundle

Format

A data.table with 5 variables, which are:

conceptUri Uniform Resource Identifier of occupations.

iscoGroup Four-level classification of occupation groups, see **ISCO**.

preferredLabel Preferred name of ESCO occupation concepts.

altLabels Alternative labels of ESCO occupation concepts.

description Description of ESCO occupation concepts.

Source

European Skills/Competences, Qualifications and Occupations **ESCO**.

 tf_idf

Term frequency–Inverse document frequency

Description

Measure weighted amount of information concerning the specificity of terms in a corpus. Term frequency–Inverse document frequency is one of the most frequently applied weighting schemes in information retrieval systems. The tf–idf is a statistical measure proportional to the number of times a word appears in the document, but is offset by the number of documents in the corpus that contain the word. Variations of the tf–idf are often used to estimate a document’s relevance given a free-text query.

Usage

```
tf_idf(
  corpus,
  stopwords = NULL,
  id_col = "id",
  text_col = "text",
  tf_weight = "double_norm",
  idf_weight = "idf_smooth",
  min_chars = 2,
  norm = TRUE
)
```

Arguments

corpus	Input data, with an id column and a text column. Can be of type data.frame or data.table.
stopwords	A character vector of stopwords. Stopwords are filtered out before calculating numerical statistics.
id_col	Input data column name with the ids of the documents.
text_col	Input data column name with the documents.
tf_weight	Weighting scheme of term frequency. Choices are raw_count, double_norm or log_norm for raw count, double normalization at 0.5 and log normalization respectively.
idf_weight	Weighting scheme of inverse document frequency. Choices are idf and idf_smooth for inverse document frequency and inverse document frequency smooth respectively.
min_chars	Words with less characters than min_chars are filtered out before calculating numerical statistics.
norm	Boolean value for document normalization.

Value

A data.table with three columns, namely class derived from given document ids, term and tfIdf.

Examples

```
library(data.table)
corpus <- copy(occupations_bundle)
invisible(corpus[, text := paste(preferredLabel, altLabels)])
invisible(corpus[, text := cleansing_corpus(text)])
corpus <- corpus[ , .(conceptUri, text)]
setnames(corpus, c("id", "text"))
tf_idf(corpus)
```

Index

* datasets

- isco_occupations_bundle, 6
- occupations_bundle, 7

amatch, 3

classify_occupation, 2

cld2, 6

cleansing_corpus, 4

data_stopwords_misc, 5

get_language_code, 5

get_stopwords, 5

identify_language, 6

isco_occupations_bundle, 3, 6

occupations_bundle, 3, 7

stopwords, 5

tf_idf, 8

tfidf, 3