Package: poldis (via r-universe)

November 3, 2024

Type Package

Title Analyse Political Texts

Version 0.1.2

Date 2024-09-03

Maintainer Henrique Sposito <henrique.sposito@graduateinstitute.ch>

Description Wrangle and annotate different types of political texts.
It also introduces Urgency Analysis, a new method for the
analysis of urgency in political texts.

URL http://henriquesposito.com/poldis/

BugReports https://github.com/henriquesposito/poldis/issues

License MIT + file LICENSE

Imports dplyr, stringr, purrr, stringi, quanteda, spacyr, textstem,
tidyr, stringdist

Suggests rmarkdown, testthat, tesseract, quanteda.textstats, keyATM,
messydates, pdftools, fmsb, ggplot2, tm, cli

RoxygenNote 7.3.2

Encoding UTF-8

LazyData True

Depends R (>= 3.5.0)

Repository https://henriquesposito.r-universe.dev

RemoteUrl https://github.com/henriquesposito/poldis

RemoteRef HEAD

RemoteSha e013060ceb0aca406970ead1143d346a1a724794

Contents

`annotate_text` *Annotate text with NLP*

Description

This function relies on '{spacyr}' NLP parsing to annotate texts.

Usage

```
annotate_text(v, level = "words")
```

Arguments

`v`	Text vector
`level`	At which level would you like to parse the text? Options include "words" or "sentences". Defaults to "words".

Value

A data frame with syntax information by words or sentences in text.

Examples

```
#annotate_text(US_News_Conferences_1960_1980[1:2, 3])
#annotate_text(US_News_Conferences_1960_1980[1:2, 3], level = "sentence")
```

`extract_context` *Extract context for string matches*

Description

A function for getting string matches and the context in which they occur.

Usage

```
extract_context(match, v, level = "sentences", n = 1)
```

Arguments

`match`	Character string to be matched. For multiple strings, please use "|" as a separator.
`v`	Text vector or annotated data frame.
`level`	At which text level do you want matches to be returned? Defaults to "sentences". Options are sentences, words, and paragraph.
`n`	Number of sentences or words matched before and after string match. Defaults to 1. That is, one word or one sentence before, and after, string match. For paragraphs, n is always set to one.

Value

A list of string matches and their context.

Examples

```
extract_context(match = "war|weapons of mass destruction|conflict|NATO|peace",
                v = US_News_Conferences_1960_1980$text[100],
                level = "sentences", n = 2)
```

`extract_date` *Extract dates from text*

Description

Wrapper function for ‘messydates::as_messydates‘.

Usage

```
extract_date(v)
```

Arguments

`v`	Text vector.

Value

A vector of the dates in text.

Examples

```
#extract_date("Today is the twenty six of February of two thousand and twenty four")
```

`extract_locations` *Extract locations from strings*

Description

Extract locations from strings

Usage

```
extract_locations(v)
```

Arguments

v	Text vector.

Details

The function relies on geographical entity detection from NLP models.

Value

A data frame of locations and the number of times they appear.

Examples

```
#extract_locations(c("This is the United States", "This is Sao Paulo",
#"I was in Rio de Janeiro and Sao Paulo, then back to the United States"))
```

`extract_match` *Extract text matches*

Description

Get texts in which certain "matches" occur.

Usage

```
extract_match(v, match, invert = FALSE, ignore.case = TRUE)
```

Arguments

`v`	Text vector or annotated data frame.
`match`	A regex match for a word(s) or expression. For multiple words, please use "|" to divide them.
`invert`	Do you want texts without certain matches to be returned? By default FALSE.
`ignore.case`	Should case be ignored? By default, TRUE.

Value

A list the same length as text variable.

Examples

```
extract_match(c("This function was created on the 29 September 2021",
"Today is October 12, 2021"), "October")
```

`extract_names` *Extract a list of possible names of individuals in texts*

Description

Extract a list of possible names of individuals in texts

Usage

```
extract_names(v)
```

Arguments

`v`	A text vector.

Details

The function relies on named entity recognition from NLP models.

Value

A data frame of individual names and the number of times they appear.

Examples

```
#extract_names(US_News_Conferences_1960_1980[20, 3])
```

`extract_text_similarities`

Extract similarities and differences in texts/segments

Description

Extract similarities and differences in texts/segments

Usage

```
extract_text_similarities(v, comparison = "similarities", method)
```

Arguments

`v`	Text vector or annotated data frame.
`comparison`	How would you like to compare texts? Options are "similarities", for comparing similarities, or "differences", for comparing differences. Defaults to "similarities".
`method`	A method for checking similarities or differences between texts. For similarities, defaults to "correlation" method. Other methods for similarities include "cosine", "jaccard", "ejaccard", "dice", "edice", "simple matching", and "hamann". For differences, defaults to "euclidean". Other methods for differences include "manhattan", "maximum", "canberra", and "minkowski". For more information on each of these methods and what are the implications in selecting a method, please see '?quanteda.textstats::textstat_simil()'.

Value

A matrix of similarity scores between texts.

Examples

```
#extract_text_similarities(US_News_Conferences_1960_1980[1:2,3])
```

`extract_title` *Extract first sentence from text*

Description

A lot of information is contained in the first sentence of a text. In political texts, for example, dates and locations are often contained in the first sentence of the text.

Usage

```
extract_title(v)
```

Arguments

`v`	Text vector.

Value

A list of the first sentences in text.

Examples

```
extract_title("This is the first sentence. This is the second sentence.")
```

`gather_related_terms` *Gather terms related to subjects*

Description

Gather terms related to subjects

Usage

```
gather_related_terms(.data, dictionary)
```

Arguments

`.data`	A data frame, priorities data frame coded using ‘select_priorities()‘, or text vector. For data frames, function will search for "text" variable. For priorities data frame function will search for "priorities" variable.
`dictionary`	The dictionary of 20 major political topics from the Comparative Agendas Project (Jones et al., 2023) is used by default. Users can also declare a custom dictionary as a vector or a list. If users declare a vector, each element is treated as a independent topic. If users declare a list of subjects and related terms, function understands names as topic and words as terms.

Details

This function relies on keyword assisted topic models implemented in the '{keyATM}' package to find related words based on the topics provided and texts in which they appear.

Value

A list of related terms to each of the topics declared in dictionary.

References

Eshima S, Imai K, and Sasaki T. 2024. "Keyword-Assisted Topic Models." _American Journal of Political Science_, 68(2): 730-750. doi:10.1111/ajps.12779

Examples

```
#gather_related_terms(US_News_Conferences_1960_1980[1:5, 3], dictionary = "CAP")
#gather_related_terms(US_News_Conferences_1960_1980[1:5, 3],
#                     dictionary = c("military", "development"))
#gather_related_terms(US_News_Conferences_1960_1980[1:5, 3],
#                     dictionary = list("military" = c("military", "gun", "war"),
#                            "development" = c("development", "interest rate", "banks")))
```

`gather_topics` *Gather topic from political discourses*

Description

Gather topic from political discourses

Usage

```
gather_topics(.data, dictionary = "CAP")
```

Arguments

`.data`	A data frame, priorities data frame coded using 'select_priorities()', or text vector. For data frames, function will search for "text" variable. For priorities data frame function will search for "priorities" variable. If missing, opens the webpage containing the political topics codebook.
`dictionary`	The dictionary of 20 major political topics from the Comparative Agendas Project (Jones et al., 2023) is used by default. Users can also declare a custom dictionary as a vector or a list. If users declare a vector, each element is treated as a independent topic. If users declare a list of subjects and related terms, function understands names as topic and words as terms. For more information on how the CAP topics were adapted, please run 'gather_topics()' to access the political topics codebook.

Value

A list of topics present in each text separated by comma.

Examples

```
gather_topics(US_News_Conferences_1960_1980[1:5, 3])
gather_topics(US_News_Conferences_1960_1980[1:5, 3],
              dictionary = c("military", "development"))
gather_topics(US_News_Conferences_1960_1980[1:5, 3],
              dictionary = list("military" = c("military", "gun", "war"),
                             "development" = c("development", "interest rate", "banks")))
#summary(gather_topics(US_News_Conferences_1960_1980[1:5, 3]))
#plot(gather_topics(US_News_Conferences_1960_1980[1:5, 3],
#                    dictionary = c("military", "development")))
```

get_urgency *Urgency Analysis*

Description

Urgency Analysis

Usage

```
get_urgency(.data, summarise = "sum")
```

Arguments

.data	A data frame, priorities data frame coded using ‘select_priorities()‘, or text vector. For data frames, function will search for "text" variable. For priorities data frame function will search for "priorities" variable. If missing, opens the webpage containing the urgency codebook.
summarise	How to handle multiple matches for the same dimension in the same text observation? By default, multiple matches are added together and their "sum" per text observation is returned. Users can, instead, choose the "mean" which returns the average score per dimension per text observation when there are multiple matches. The "mean" can also be used as a form of normalization per dimension and text observation in certain cases.

Details

Urgency in political discourses is an expression of how necessary and/or how soon an action should be undertaken or completed. This is measured along four dimensions, two related to necessity and two related to timing. The first two dimensions, degree of intensity and degree of commitment, relate to the necessity of taking the action, while the next two dimensions, frequency of action and timing of action, relate to the timing in which action is taken. Our dictionary includes terms in each of these dimensions. The terms included in each of these dimensions were validated and adjusted through an online survey that took place between July and August of 2024. The survey results were recorded as counts of the number of participants who selected an urgency-related word as more urgent than its pair. To analyze the survey results, we employed Bradley-Terry models for paired comparisons. A rank of the words for each dimension of urgency was obtained from the analysis, which were then used to create the urgency word scores in the dictionaries. For more information on the dimensions, scores, or the survey on urgency, please run 'get_urgency()' to access the urgency codebook. For priorities (i.e. coded using the 'select_priorities()'), urgency scores are calculated by multiplying the commitment scores by all other dimensions. This is done because commitment words are indicative of political priorities, For more information please refer to the 'select_priorities()' function. For vectors or data frames urgency scores are calculated by adding commitment and intensity dimension scores (i.e. how necessary) and multiplying these by the sum of timing and frequency dimension scores (i.e. how soon). In both cases, zero urgency scores are indicative of no urgency but maximum scores can vary.

Value

A scored data frame for each dimension of urgency.

Examples

```
get_urgency(US_News_Conferences_1960_1980[1:10, 3])
get_urgency(US_News_Conferences_1960_1980[1:10,])
#get_urgency(select_priorities(US_News_Conferences_1960_1980[1:2, 3]))
#summary(get_urgency(US_News_Conferences_1960_1980[1:10, 3]))
#plot(get_urgency(US_News_Conferences_1960_1980[1:10, 3]))
#get_urgency()
```

`read_pdf` *Read text from PDFs*

Description

Read text from PDFs

Usage

```
read_pdf(path)
```

Arguments

`path`	The path to a PDF file or a folder containing multiple PDFs.

Value

A list of texts.

`select_priorities` *Select future priorities from political discourses*

Description

Political priorities are statements in which actors express their intent or commitment to take political action in the future.

Usage

```
select_priorities(.data, na.rm = TRUE)
```

Arguments

`.data`	A (annotated) data frame or text vector. For data frames, function will search for "text" variable. For annotated data frames, please declare an annotated data frame at the sentence level.
`na.rm`	Would you like political statements that do not contain a political action to be removed? By default, TRUE.

Value

A data frame with syntax information by sentences and a variable identifying which of these sentences are priorities.

Examples

```
#select_priorities(US_News_Conferences_1960_1980[1:2,3])
```

`sim_urgency` *Simulating urgency in priorities*

Description

Simulating urgency in priorities

Usage

```
sim_urgency(urgency, commitment, intensity, timing, frequency, pronoun = "We")
```

Arguments

`urgency`	Desired urgency score, optional.
`commitment`	Desired commitment score, optional.
`intensity`	Desired intensity score, optional.
`timing`	Desired timing score, optional.
`frequency`	Desired frequency score, optional.
`pronoun`	How would you like the simulated priorities to start? By default, priorities start with the pronoun "We".

Details

Users can declare a score for one or more of the urgency dimensions or an urgency score. This means, if users may not declare an urgency score and the score for one or more dimensions at once. In those cases, the urgency score is favored.

Value

A sentence that matches the urgency or urgency dimension scores.

Examples

```
sim_urgency()
sim_urgency(urgency = 0.5)
sim_urgency(urgency = 2.5)
sim_urgency(urgency = -2.5)
sim_urgency(commitment = 0.6)
sim_urgency(commitment = 0.6, intensity = 1.4)
sim_urgency(commitment = 0.6, intensity = 1.4, timing = 1.4)
sim_urgency(commitment = 0.6, intensity = 1.2, timing = 1.4, frequency = 1.8)
```

`split_text` *Split texts*

Description

Split texts into structured lists of lists according to a split sign.

Usage

```
split_text(v, splitsign = "\\.")
```

Arguments

`v`	Text vector or annotated data frame.
`splitsign`	Where do you want to split? By default sentences ("."). This can also be words, signals or other markers you want. For special characters, please use escape sign before (i.e. "\").

Value

A list of lists the same length as vector.

Examples

```
split_text("This is the first sentence. This is the second sentence.")
```

`US_News_Conferences_1960_1980`
US News Conferences Data from 1960 to 1980

Description

A dataset containing the news conferences from US presidents from 1960 to 1980. The dataset was gathered from the American Presidency Project website.

Usage

```
data(US_News_Conferences_1960_1980)
```

Format

A data frame with 353 rows and 3 variables: the president, the date, and the full text.

Index