# Package ‘syuzhet’

November 24, 2020

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Extracts Sentiment and Sentiment-Derived Plot Arcs from Text</td>
</tr>
<tr>
<td>Version</td>
<td>1.0.6</td>
</tr>
<tr>
<td>Date</td>
<td>2020-10-29</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Matthew Jockers <a href="mailto:mjockers@gmail.com">mjockers@gmail.com</a></td>
</tr>
<tr>
<td>Description</td>
<td>Extracts sentiment and sentiment-derived plot arcs from text using a variety of sentiment dictionaries conveniently packaged for consumption by R users. Implemented dictionaries include <code>syuzhet'' (default) developed in the Nebraska Literary Lab, </code>afinn'' developed by Finn Årup Nielsen, <code>bing'' developed by Minqing Hu and Bing Liu, and </code>nrc'' developed by Mohammad, Saif M. and Turney, Peter D. Applicable references are available in README.md and in the documentation for the ``get_sentiment'' function. The package also provides a hack for implementing Stanford's coreNLP sentiment parser. The package provides several methods for plot arc normalization.</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://github.com/mjockers/syuzhet">https://github.com/mjockers/syuzhet</a></td>
</tr>
<tr>
<td>License</td>
<td>GPL-3</td>
</tr>
<tr>
<td>Depends</td>
<td>R (&gt;= 3.5.0)</td>
</tr>
<tr>
<td>Imports</td>
<td>textshape (&gt;= 1.3.0), NLP, zoo, dtt, stats, graphics, dplyr, tidyr, rlang</td>
</tr>
<tr>
<td>LazyData</td>
<td>true</td>
</tr>
<tr>
<td>Suggests</td>
<td>devtools, knitr, pander, parallel, readxl, rmarkdown, stringr, testthat (&gt;= 0.9.1)</td>
</tr>
<tr>
<td>NeedsCompilation</td>
<td>no</td>
</tr>
<tr>
<td>VignetteBuilder</td>
<td>knitr</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.1.1</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Author</td>
<td>Matthew Jockers [aut, cre]</td>
</tr>
<tr>
<td>Repository</td>
<td>CRAN</td>
</tr>
<tr>
<td>Date/Publication</td>
<td>2020-11-24 15:50:02 UTC</td>
</tr>
</tbody>
</table>
R topics documented:

- `get_dct_transform` ............................................. 2
- `get_nrc_sentiment` ........................................... 3
- `get_nrc_values` ............................................. 4
- `get_percentage_values` ..................................... 4
- `get_sentences` ............................................... 5
- `get_sentiment` ............................................... 6
- `get_sentiment_dictionary` .................................. 7
- `get_sent_values` ............................................. 8
- `get_stanford_sentiment` .................................... 8
- `get_text_as_string` ......................................... 9
- `get_tokens` ................................................ 9
- `get_transformed_values` .................................... 10
- `mixed_messages` ............................................ 11
- `rescale` ................................................... 12
- `rescale_x_2` ............................................... 12
- `simple_plot` ............................................... 13

Index 14

---

`get_dct_transform` *Discrete Cosine Transformation with Reverse Transform.*

**Description**

Converts input values into a standardized set of filtered and reverse transformed values for easy plotting and/or comparison.

**Usage**

```r
get_dct_transform(
  raw_values,
  low_pass_size = 5,
  x_reverse_len = 100,
  scale_vals = FALSE,
  scale_range = FALSE
)
```

**Arguments**

- `raw_values` the raw sentiment values calculated for each sentence
- `low_pass_size` The number of components to retain in the low pass filtering. Default = 5
- `x_reverse_len` the number of values to return via decimation. Default = 100
- `scale_vals` Logical determines whether or not to normalize the values using the scale function Default = FALSE. If TRUE, values will be scaled by subtracting the means and scaled by dividing by their standard deviations. See ?scale
scale_range Logical determines whether or not to scale the values from -1 to +1. Default = FALSE. If set to TRUE, the lowest value in the vector will be set to -1 and the highest values set to +1 and all the values scaled accordingly in between.

Value
The transformed values

Examples
s_v <- get_sentences("I begin this story with a neutral statement. Now I add a statement about how much I despise cats. I am allergic to them. I hate them. Basically this is a very silly test. But I do love dogs!")
raw_values <- get_sentiment(s_v, method = "syuzhet")
dct_vals <- get_dct_transform(raw_values)
plot(dct_vals, type="l", ylim=c(-0.1,.1))
get_nrc_values

Summarize NRC Values

Description
Access the NRC dictionary to compute emotion types and valence for a set of words in the input vector.

Usage
get_nrc_values(word_vector, language = "english", lexicon = NULL)

Arguments
- word_vector: A character vector.
- language: A string
- lexicon: A data frame with at least the columns "word", "sentiment" and "value". If NULL, internal data will be taken.

Value
A vector of values for the emotions and valence detected in the input vector.

get_percentage_values

Chunk a Text and Get Means

Description
Chunks text into 100 Percentage based segments and calculates means.

Usage
get_percentage_values(raw_values, bins = 100)

Arguments
- raw_values: Raw sentiment values
- bins: The number of bins to split the input vector. Default is 100 bins.

Value
A vector of mean values from each chunk
**get_sentences**  

Sentence Tokenization

**Description**

Parses a string into a vector of sentences.

**Usage**

get_sentences(text_of_file, fix_curly_quotes = TRUE, as_vector = TRUE)

**Arguments**

- **text_of_file** A Text String
- **fix_curly_quotes** logical. If TRUE curly quotes will be converted to ASCII representation before splitting.
- **as_vector** If TRUE the result is unlisted. If FALSE the result stays as a list of the original text string elements split into sentences.

**Value**

A Character Vector of Sentences

**Examples**

(x <- c(paste0(  
  "Mr. Brown comes! He says hello. i give him coffee. i will ",  
  "go at 5 p. m. eastern time. Or somewhere in between!go there"
  ),  
  paste0(  
    "Marvin K. Mooney Will You Please Go Now!", "The time has come.",  
    "The time has come. The time is now. Just go. Go. GO!",  
    "I don't care how."
  )))

get_sentences(x)
get_sentences(x, as_vector = FALSE)
get_sentiment

Get Sentiment Values for a String

Description

Iterates over a vector of strings and returns sentiment values based on user supplied method. The
default method, "syuzhet" is a custom sentiment dictionary developed in the Nebraska Literary
Lab. The default dictionary should be better tuned to fiction as the terms were extracted from a
collection of 165,000 human coded sentences taken from a small corpus of contemporary novels.
At the time of this release, Syuzhet will only work with languages that use Latin character sets. This
"Thai", "Ukrainian", "Urdu", "Yiddish" are not supported even though these languages are part of
the extended NRC dictionary.

Usage

get_sentiment(
  char_v,
  method = "syuzhet",
  path_to_tagger = NULL,
  cl = NULL,
  language = "english",
  lexicon = NULL,
  regex = "[^A-Za-z']+",
  lowercase = TRUE
)

Arguments

char_v A vector of strings for evaluation.
method A string indicating which sentiment method to use. Options include "syuzhet",
"bing", "afinn", "nrc" and "stanford." See references for more detail on methods.
path_to_tagger local path to location of Stanford CoreNLP package
cl Optional, for parallel sentiment analysis.
language A string. Only works for "nrc" method
lexicon a data frame with at least two columns labeled "word" and "value."
regex A regular expression for splitting words. Default is "[^A-Za-z']+"
lowercase should tokens be converted to lowercase. Default equals TRUE

Value

Return value is a numeric vector of sentiment values, one value for each input sentence.
get_sentiment_dictionary

References


get_sentiment_dictionary

Sentiment Dictionaries

Description

Get the sentiment dictionaries used in syuzhet.

Usage

get_sentiment_dictionary(dictionary = "syuzhet", language = "english")

Arguments

dictionary A string indicating which sentiment dictionary to return. Options include "syuzhet", "bing", "afinn", and "nrc".

language A string indicating the language to choose if using the NRC dictionary and a language other than English
get_sentiment_dictionary

Assigns sentiment values to words based on preloaded dictionary. The default is the Syuzhet dictionary.

Usage

get_sentiment_dictionary()
get_sentiment_dictionary('bing')
get_sentiment_dictionary('afinn')
get_sentiment_dictionary('nrc', language = 'spanish')

get_sent_values

Assigns Sentiment Values

Description

Assigns sentiment values to words based on preloaded dictionary. The default is the Syuzhet dictionary.

Usage

get_sent_values(char_v, method = 'syuzhet', lexicon = NULL)

Arguments

cchar_v A string
method A string indicating which sentiment dictionary to use
lexicon A data frame with at least two columns named word and value. Works with "nrc" or "custom" method. If using custom method, you must load a custom lexicon as a data frame with aforementioned columns.

Value

A single numerical value (positive or negative) based on the assessed sentiment in the string

get_stanford_sentiment

Get Sentiment from the Stanford Tagger

Description

Call the Stanford Sentiment tagger with a vector of strings. The Stanford tagger automatically detects sentence boundaries and treats each sentence as a distinct instance to measure. As a result, the vector that gets returned will not be the same length as the input vector.
### get_text_as_string

**Usage**

```r
get_text_as_string(path_to_file)
```

**Description**

Loads a file as a single text string.

**Arguments**

- `path_to_file`: file path

**Value**

A character vector of length 1 containing the text of the file in the path_to_file argument.

### get_tokens

**Usage**

```r
get_tokens(text_of_file, pattern = "\\w", lowercase = TRUE)
```

**Description**

Parses a string into a vector of word tokens.

**Arguments**

- `text_of_file`: A Text String
- `pattern`: A regular expression for token breaking
- `lowercase`: should tokens be converted to lowercase. Default equals TRUE

**Value**

A Character Vector of Words
get_transformed_values

Fourier Transform and Reverse Transform Values

Description

Please Note: This function is maintained for legacy purposes. Users should consider using get_dct_transform() instead. Converts input values into a standardized set of filtered and reverse transformed values for easy plotting and/or comparison.

Usage

get_transformed_values(
  raw_values,
  low_pass_size = 2,
  x_reverse_len = 100,
  padding_factor = 2,
  scale_vals = FALSE,
  scale_range = FALSE
)

Arguments

raw_values the raw sentiment values calculated for each sentence
low_pass_size The number of components to retain in the low pass filtering. Default = 3
x_reverse_len the number of values to return. Default = 100
padding_factor the amount of zero values to pad raw_values with, as a factor of the size of raw_values. Default = 2.
scale_vals Logical determines whether or not to normalize the values using the scale function. Default = FALSE. If TRUE, values will be scaled by subtracting the means and scaled by dividing by their standard deviations. See ?scale
scale_range Logical determines whether or not to scale the values from -1 to +1. Default = FALSE. If set to TRUE, the lowest value in the vector will be set to -1 and the highest values set to +1 and all the values scaled accordingly in between.

Value

The transformed values

Examples

s_v <- get_sentences("I begin this story with a neutral statement. Now I add a statement about how much I despise cats. I am allergic to them. Basically this is a very silly test.")
raw_values <- get_sentiment(s_v, method = "bing")
get_transformed_values(raw_values)
mixed_messages

Description

This function calculates the "emotional entropy" of a string based on the amount of conflicting valence. Emotional entropy is a measure of unpredictability and surprise based on the consistency or inconsistency of the emotional language in a given string. A string with conflicting emotional language may be said to express a "mixed message."

Usage

mixed_messages(string, removeNeutral = TRUE)

Arguments

string A string of words
removeNeutral Logical indicating whether or not to remove words with neutral valence before computing the emotional entropy of the string. Default is TRUE

Value

A vector containing two named values

Examples

text_v <- "That's the love and the hate of it"
mixed_messages(text_v) # [1] 1.0 0.5 = high (1.0, 0.5) entropy
mixed_messages(text_v, TRUE)
# Example of a predictable message i.e. no surprise
  text_v <- "I absolutely love, love, love it."
mixed_messages(text_v) # [1] 0 0 = low entropy e.g. totally consistent emotion, i.e. no surprise
mixed_messages(text_v, FALSE)
# A more realistic example with a lot of mixed emotion.
  text_v <- "I loved the way he looked at me but I hated that he was no longer my lover"
mixed_messages(text_v) # [1] 0.91829583 0.05101644 pretty high entropy.
mixed_messages(text_v, FALSE)
# A more realistic example without a lot of mixed emotion.
  text_v <- "I loved the way he looked at me and I was happy that he was my lover."
mixed_messages(text_v) # [1] 0 0 low entropy, no surprise.
mixed_messages(text_v, FALSE)
# An unrealistic example with a lot of mixed emotion.
  text_v <- "I loved, hated and despised the way he looked at me and I was happy as hell that he was my white hot lover."
mixed_messages(text_v)
mixed_messages(text_v, FALSE)
**rescale**

*Vector Value Rescaling*

**Description**

Rescale Transformed values from -1 to 1

**Usage**

```
rescale(x)
```

**Arguments**

- **x**: A vector of values

---

**rescale_x_2**

*Bi-Directional x and y axis Rescaling*

**Description**

Rescales input values to two scales (0 to 1 and -1 to 1) on the y-axis and also creates a scaled vector of x axis values from 0 to 1. This function is useful for plotting and plot comparison.

**Usage**

```
rescale_x_2(v)
```

**Arguments**

- **v**: A vector of values

**Value**

A list of three vectors (x, y, z). x is a vector of values from 0 to 1 equal in length to the input vector v. y is a scaled (from 0 to 1) vector of the input values equal in length to the input vector v. z is a scaled (from -1 to +1) vector of the input values equal in length to the input vector v.
simple_plot

Plots simple and rolling shapes overlayed

**Description**

A simple function for comparing three smoothers

**Usage**

```r
simple_plot(
  raw_values,
  title = "Syuzhet Plot",
  legend_pos = "top",
  lps = 10,
  window = 0.1
)
```

**Arguments**

- `raw_values` the raw sentiment values calculated for each sentence
- `title` for resulting image
- `legend_pos` position for legend
- `lps` size of the low pass filter. I.e. the number of low frequency components to retain
- `window` size of the rolling window for the rolling mean expressed as a percentage.
Index

data.frame, 8
get_dct_transform, 2
get_nrc_sentiment, 3
get_nrc_values, 4
get_percentage_values, 4
get_sent_values, 8
get_sentences, 5
get_sentiment, 6
get_sentiment_dictionary, 7
get_stanford_sentiment, 8
get_text_as_string, 9
get_tokens, 9
get_transformed_values, 10
mixed_messages, 11
rescale, 12
rescale_x_2, 12
simple_plot, 13
vector, 11