Package: syuzhet (via r-universe)

November 8, 2024

Type Package

Title Extracts Sentiment and Sentiment-Derived Plot Arcs from Text

Version 1.0.7

Date 2023-8-11

Maintainer Matthew Jockers <mjockers@gmail.com>

Description 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 ``syuzhet'' (default) developed in the Nebraska Literary Lab ``afinn'' developed by Finn Årup Nielsen, ``bing'' developed by Minqing Hu and Bing Liu, and ``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.

URL https://github.com/mjockers/syuzhet

License GPL-3

Depends R (>= 3.5.0)

Imports textshape (>= 1.3.0), NLP, zoo, dtt, stats, graphics, dplyr, tidyr, rlang

Suggests devtools, knitr, pander, parallel, readxl, rmarkdown, stringr, testthat (>= 0.9.1)

NeedsCompilation no

VignetteBuilder knitr

RoxygenNote 7.2.3

Encoding UTF-8

Config/pak/sysreqs libicu-dev

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

RemoteUrl https://github.com/mjockers/syuzhet

RemoteRef HEAD

RemoteSha 73ec7d852e1f368661069e1b75e6c8c12a9f2133

Contents

---

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

```
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_sentiment` *Get Emotions and Valence from NRC Dictionary*

Description

Calls the NRC sentiment dictionary to calculate the presence of eight different emotions and their corresponding valence in a text file.

Usage

```
get_nrc_sentiment(
  char_v,
  cl = NULL,
  language = "english",
  lowercase = TRUE,
  lexicon = NULL
)
```

Arguments

`char_v`	A character vector
`cl`	Optional, for parallel analysis
`language`	A string
`lowercase`	should tokens be converted to lowercase. Default equals TRUE
`lexicon`	a custom lexicon provided by the user and formatted as a data frame containing two columns labeled as "word" and "sentiment". The "sentiment" column must indicate either the valence of the word (using either the term "positive" or "negative") or the emotional category of the word, using one of the following terms: "anger", "anticipation", "disgust", "fear", "joy", "sadness", "surprise", "trust". For example: the English word "abandon" may appear in your lexicon twice,

first with a emotional category of "fear" and again with a value of "negative." Not all words necessarily need to have a valence indicator. See example section below

Value

A data frame where each row represents a sentence From the original file. The columns include one for each emotion type as well as a positive or negative valence. The ten columns are as follows: "anger", "anticipation", "disgust", "fear", "joy", "sadness", "surprise", "trust", "negative", "positive."

References

Saif Mohammad and Peter Turney. "Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon." In Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, June 2010, LA, California. See: http://saifmohammad.com/WebPages/lexicons.html

Examples

```
my_lexicon <- data.frame(
word = c("love","love", "hate", "hate"),
sentiment = c("positive", "joy", "negative", "anger")
)
my_example_text <- "I am in love with R programming.
  I hate writing code in C."
s_v <- get_sentences(my_example_text)
get_nrc_sentiment(s_v, lexicon=my_lexicon)
```

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 effectively means that "Arabic", "Bengali", "Chinese_simplified", "Chinese_traditional", "Greek", "Gujarati", "Hebrew", "Hindi", "Japanese", "Marathi", "Persian", "Russian", "Tamil", "Telugu", "Thai", "Ukranian", "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.

References

Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.

Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA. See: http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon

Saif Mohammad and Peter Turney. "Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon." In Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, June 2010, LA, California. See: http://saifmohammad.com/WebPages/lexicons.html

Finn Årup Nielsen. "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs", Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts':Big things come in small packages 718 in CEUR Workshop Proceedings : 93-98. 2011 May. http://arxiv.org/abs/1103.2903. See: http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60. See: http://nlp.stanford.edu/software/corenlp.shtml

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng and Christopher Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank Conference on Empirical Methods in Natural Language Processing" (EMNLP 2013). See: http://nlp.stanford.edu/sentiment/

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

Value

A `data.frame`

Examples

```
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

`char_v`	A string
`method`	A string indicating which sentiment dictionary to use
`lexicon`	A data frame with 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 aforementioend 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.

Usage

```
get_stanford_sentiment(text_vector, path_to_stanford_tagger)
```

Arguments

`text_vector`	A vector of strings
`path_to_stanford_tagger`	a local file path indicating where the coreNLP package is installed.

`get_text_as_string` *Load Text from a File*

Description

Loads a file as a single text string.

Usage

```
get_text_as_string(path_to_file)
```

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 *Word Tokenization*

Description

Parses a string into a vector of word tokens.

Usage

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

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` *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, remove_neutral = TRUE)
```

Arguments

`string` A string of words

`remove_neutral` 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 absolutley 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 urealistic 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

```
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