Package 'SentimentAnalysis'

June 16, 2016

Type Package

Title Dictionary-based sentiment analysis

Version 1.0-0

Date 2016-06-26

Author Stefan Feuerriegel [aut, cre],
Nicolas Proellochs [aut]

Maintainer Stefan Feuerriegel <stefan.feuerriegel@is.uni-freiburg.de>

Description Performs a sentiment analysis of textual contents in R. This implementation utilizes various existing dictionaries, such as General Inquirer, Harvard IV or Loughran-McDonald. Furthermore, it can also create customized dictionaries. The latter uses LASSO regularization as a statistical approach to select relevant terms based on an exogeneous response variable.

License MIT + file LICENSE

Depends R (>= 2.10)

Imports tm (>= 0.6),
qdapDictionaries,
ngramrr (>= 0.1),
moments,
stringdist,
SnowballC,
XML,
glmnet

Suggests testthat,
knitr,
rmarkdown

LazyData true

RoxygenNote 5.0.1

VignetteBuilder knitr

R topics documented:

`analyzeSentiment` *Sentiment analysis*

Description

Performs sentiment analysis of given object (vector of strings, document-term matrix, corpus).

Usage

```
analyzeSentiment(x, language, aggregate, ...)

## S3 method for class 'Corpus'
analyzeSentiment(x, language = "english", aggregate, ...)

## S3 method for class 'character'
analyzeSentiment(x, language = "english", aggregate, ...)

## S3 method for class 'data.frame'
analyzeSentiment(x, language = "english", aggregate, ...)

## S3 method for class 'TermDocumentMatrix'
analyzeSentiment(x, language = "english",
  aggregate, ...)

## S3 method for class 'DocumentTermMatrix'
```

```
analyzeSentiment(x, language = "english",
  aggregate, ...)
```

Arguments

x	A vector of characters, a data.frame, an object of type Corpus, TermDocumentMatrix or DocumentTermMatrix
language	Language used for preprocessing operations (default: English)
aggregate	A factor variable by which documents can be grouped. This helpful when joining e.g. news from the same day or move reviews by the same author
...	Additional parameters passed to function for e.g. preprocessing

Details

This function returns a data.frame with continuous values. If one desires other formats, one needs to convert these. Common examples of such formats are binary response values (positive / negative) or tertiary (positive, neutral, negative). Hence, consider using the functions convertToBinaryResponse and convertToDirection, which can convert a vector of continuous sentiment scores into a factor object.

Value

Result is a matrix which sentiment values for each document across all defined rules

See Also

compareToResponse, convertToBinaryResponse, convertToDirection

Examples

```
# via vector of strings
corpus <- c("Positive text", "Neutral but uncertain text", "Negative text")
sentiment <- analyzeSentiment(corpus)
compareToResponse(sentiment, c(+1, 0, -1))

# via Corpus from tm package
library(tm)
reut21578 <- system.file("texts", "crude", package="tm")
reuters <- Corpus(DirSource(reut21578),
                  readerControl=list(reader=readReut21578XML))

# via DocumentTermMatrix (with stemmed entries)
dtm <- DocumentTermMatrix(Corpus(VectorSource(c("posit posit", "negat neutral"))))
sentiment <- analyzeSentiment(dtm)
compareToResponse(sentiment, c(TRUE, FALSE))
```

compareDictionaries *Compares two dictionaries*

Description

Routine compares two dictionaries in terms of how similarities and differences. Among the calculated measures are the total of distinct words, the overlap between both dictionaries, etc.

Usage

```
compareDictionaries(d1, d2)
```

Arguments

d1	is the first sentiment dictionary of type SentimentDictionaryWordlist, SentimentDictionaryBin or SentimentDictionaryWeighted
d2	is the first sentiment dictionary of type SentimentDictionaryWordlist, SentimentDictionaryBin or SentimentDictionaryWeighted

Value

Returns list with different metrics depending on dictionary type

Note

Currently, this routine only supports the case where both dictionaries are of the same type

See Also

SentimentDictionaryWordlist SentimentDictionaryBinary SentimentDictionaryWeighted

Examples

```
d1 <- SentimentDictionary(c("uncertain", "possible", "likely"))
d2 <- SentimentDictionary(c("rather", "intend", "likely"))
cmp <- compareDictionaries(d1, d2)

d1 <- SentimentDictionary(c("increase", "rise", "more"),
                          c("fall", "drop"))
d2 <- SentimentDictionary(c("positive", "rise", "more"),
                          c("negative", "drop"))
cmp <- compareDictionaries(d1, d2)

d1 <- SentimentDictionary(c("increase", "decrease", "exit"),
                          c(+1, -1, -10),
                          rep(NA, 3))
d2 <- SentimentDictionary(c("increase", "decrease", "drop", "neutral"),
                          c(+2, -5, -1, 0),
                          rep(NA, 4))
cmp <- compareDictionaries(d1, d2)
```

compareToResponse *Compare sentiment values to existing response variable*

Description

This function compares the calculated sentiment values with an external response variable. Examples of such an exogenous response are stock market movements or IMDb move rating. Both usually reflect a "true" value that the sentiment should match.

Usage

```
compareToResponse(sentiment, response)

## S3 method for class 'logical'
compareToResponse(sentiment, response)

## S3 method for class 'factor'
compareToResponse(sentiment, response)

## S3 method for class 'integer'
compareToResponse(sentiment, response)

## S3 method for class 'data.frame'
compareToResponse(sentiment, response)

## S3 method for class 'numeric'
compareToResponse(sentiment, response)
```

Arguments

sentiment	Matrix with sentiment scores for each document across several sentiment rules
response	Vector with "true" response. This vector can either be of a continuous numeric or binary values. In case of the latter, FALSE is matched to a negative sentiment value, while TRUE is matched to a non-negative one.

Value

Matrix with different performance metrics for all given sentiment rules

Examples

```
sentiment <- matrix(c(5.5, 2.9, 0.9, -1),
                    dimnames=list(c("A", "B", "C", "D"), c("Sentiment")))

# continuous numeric response variable
response <- c(5, 3, 1, -1)
compareToResponse(sentiment, response)

# binary response variable
response <- c(TRUE, TRUE, FALSE, FALSE)
compareToResponse(sentiment, response)
```

`convertToBinaryResponse` *Convert continuous sentiment to direction*

Description

This function converts continuous sentiment scores into a their corresponding binary sentiment class. As such, the result is a factor with two levels indicating positive and negative content. Neutral documents (with a sentiment score of 0) are counted as positive.

Usage

```
convertToBinaryResponse(sentiment)
```

Arguments

`sentiment`	Vector, matrix or data.frame with sentiment scores.

Details

If a matrix or data.frame is provided, this routine does not touch all columns. In fact, it scans for those where the colum name starts with "Sentiment" and changes these columns only. Hence, colums with pure negativity, positivity or ratios or word counts are ignored.

Value

If a vector is supplied, it returns a factor with two levels representing positive and negative content. Otherwise, it returns a data.frame with the corresponding columsn being exchanged.

See Also

`convertToDirection`

Examples

```
sentiment <- c(-1, -0.5, +1, 0.6, 0)
convertToBinaryResponse(sentiment)
convertToDirection(sentiment)

df <- data.frame(No=1:5, Sentiment=sentiment)
df
convertToBinaryResponse(df)
convertToDirection(df)
```

convertToDirection *Convert continuous sentiment to direction*

Description

This function converts continuous sentiment scores into a their corresponding sentiment direction. As such, the result is a factor with three levels indicating positive, neutral and negative content. In contrast to convertToBinaryResponse, neutral documents have their own category.

Usage

```
convertToDirection(sentiment)
```

Arguments

sentiment	Vector, matrix or data.frame with sentiment scores.

Details

If a matrix or data.frame is provided, this routine does not touch all columns. In fact, it scans for those where the colum name starts with "Sentiment" and changes these columns only. Hence, colums with pure negativity, positivity or ratios or word counts are ignored.

Value

If a vector is supplied, it returns a factor with three levels representing positive, neutral and negative content. Otherwise, it returns a data.frame with the corresponding columsn being exchanged.

See Also

convertToBinaryResponse

Examples

```
sentiment <- c(-1, -0.5, +1, 0.6, 0)
convertToBinaryResponse(sentiment)
convertToDirection(sentiment)

df <- data.frame(No=1:5, Sentiment=sentiment)
df
convertToBinaryResponse(df)
convertToDirection(df)
```

DictionaryGI — *Dictionary with opinionated words from the Harvard-IV dictionary as used in the General Inquirer software*

Description

Dictionary with a list of positive and negative words according to the psychological Harvard-IV dictionary as used in the General Inquirer software. This is a general-purpose dictionary developed by the Harvard University.

Usage

```
data(DictionaryGI)
```

Format

A list with different terms according to Henry

Note

All words are in lower case and non-stemmed

Source

http://www.wjh.harvard.edu/~inquirer/

Examples

```
data(DictionaryGI)
summary(DictionaryGI)
```

DictionaryHE — *Dictionary with opinionated words from Henry's Financial dictionary*

Description

Dictionary with a list of positive and negative words according to the Henry's finance-specific dictionary. This dictionary was first presented in the *Journal of Business Communication* among one of the early adopters of text analysis in the finance discipline.

Usage

```
data(DictionaryHE)
```

Format

A list with different wordlists according to Henry

Note

All words are in lower case and non-stemmed

References

Henry (2008) *Are Investors Influenced By How Earnings Press Releases Are Written?*, Journal of Business Communication, 45:4, 363-407

Examples

```
data(DictionaryHE)
summary(DictionaryHE)
```

`DictionaryLM` *Dictionary with opinionated words from Loughran-McDonald Financial dictionary*

Description

Dictionary with a list of positive, negative and uncertainty words according to the Loughran-McDonald finance-specific dictionary. This dictionary was first presented in the *Journal of Finance* and has been widely used in the finance domain ever since.

Usage

```
data(DictionaryLM)
```

Format

A list with different terms according to Loughran-McDonald

Note

All words are in lower case and non-stemmed

Source

http://www3.nd.edu/~mcdonald/Word_Lists.html

References

Loughran and McDonald (2011) *When is a Liability not a Liability? Textual Analysis, Dictionaries, and 10-Ks*, Journal of Finance, 66:1, 35-65

Examples

```
data(DictionaryLM)
summary(DictionaryLM)
```

loadDictionaryGI *Loads Harvard-IV dictionary into object*

Description

Loads Harvard-IV dictionary (as used in General Inquirer) into a standardized dictionary object

Usage

```
loadDictionaryGI()
```

Value

object of class SentimentDictionary

Note

Result is a list of stemmed words in lower case

loadDictionaryHE *Loads Henry's finance-specific dictionary into object*

Description

Loads Henry's finance-specific dictionar into a standardized dictionary object

Usage

```
loadDictionaryHE()
```

Value

object of class SentimentDictionary

Note

Result is a list of stemmed words in lower case

`loadDictionaryLM` *Loads Loughran-McDonald dictionary into object*

Description

Loads Loughran-McDonald financial dictionary into a standardized dictionary object (here, categories positive and negative are considered)

Usage

```
loadDictionaryLM()
```

Value

object of class `SentimentDictionary`

Note

Result is a list of stemmed words in lower case

`loadDictionaryLM_Uncertainty` *Loads uncertainty words from Loughran-McDonald into object*

Description

Loads uncertainty words from Loughran-McDonald into a standardized dictionary object

Usage

```
loadDictionaryLM_Uncertainty()
```

Value

object of class `SentimentDictionary`

Note

Result is a list of stemmed words in lower case

loadDictionaryQDAP *Loads polarity words from qdap package into object*

Description

Loads polarity words from data object `key.pol` which is by the package qdap. This is then converted into a standardized dictionary object

Usage

```
loadDictionaryQDAP()
```

Value

object of class `SentimentDictionary`

Note

Result is a list of stemmed words in lower case

Source

https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

References

Hu and Liu (2004). Mining Opinion Features in Customer Reviews. National Conference on Artificial Intelligence.

ngram_tokenize *N-gram tokenizer*

Description

A tokenizer for use with a document-term matrix from the tm package. Supports both character and word ngrams, including own wrapper to handle non-Latin encodings

Usage

```
ngram_tokenize(x, char = FALSE, ngmin = 1, ngmax = 3)
```

Arguments

`x`	input string
`char`	boolean value specifying whether to use character (char = TRUE) or word n-grams (char = FALSE, default)
`ngmin`	integer giving the minimum order of n-gram (default: 1)
`ngmax`	integer giving the maximum order of n-gram (default: 3)

Examples

```
library(tm)
en <- c("Romeo loves Juliet", "Romeo loves a girl")
en.corpus <- Corpus(VectorSource(en))
tdm <- TermDocumentMatrix(en.corpus,
                          control=list(wordLengths=c(1,Inf),
                                       tokenize=function(x) ngram_tokenize(x, char=TRUE,
                                                                           ngmin=3, ngmax=3)))
inspect(tdm)

ch <- c("abab", "aabb")
ch.corpus <- Corpus(VectorSource(ch))
tdm <- TermDocumentMatrix(ch.corpus,
                          control=list(wordLengths=c(1,Inf),
                                       tokenize=function(x) ngram_tokenize(x, char=TRUE,
                                                                           ngmin=1, ngmax=2)))
inspect(tdm)
```

---

numEntries *Number of words in dictionary*

Description

Counts total number of entries in dictionary.

Usage

```
numEntries(d)
```

Arguments

d	Dictionary of type SentimentDictionaryWordlist, SentimentDictionaryBinary or SentimentDictionaryWeighted

See Also

numPositiveEntries numNegativeEntries

Examples

```
numEntries(SentimentDictionary(c("uncertain", "possible", "likely"))) # returns 3
numEntries(SentimentDictionary(c("increase", "rise", "more"),
                               c("fall", "drop"))) # returns 5
numEntries(SentimentDictionary(c("increase", "decrease", "exit"),
                                 c(+1, -1, -10),
                                 rep(NA, 3))) # returns 3
```

numNegativeEntries *Number of negative words in dictionary*

Description

Counts total number of negative entries in dictionary.

Usage

```
numNegativeEntries(d)
```

Arguments

d	is a dictionary of type SentimentDictionaryBinary or SentimentDictionaryWeighted

Note

Entries in SentimentDictionaryWeighted with a weight of 0 are not counted here

See Also

numEntries numPositiveEntries

Examples

```
numNegativeEntries(SentimentDictionary(c("increase", "rise", "more"),
                           c("fall", "drop"))) # returns 2
numNegativeEntries(SentimentDictionary(c("increase", "decrease", "exit"),
                              c(+1, -1, -10),
                              rep(NA, 3))) # returns 2
```

numPositiveEntries *Number of positive words in dictionary*

Description

Counts total number of positive entries in dictionary.

Usage

```
numPositiveEntries(d)
```

Arguments

d	is a dictionary of type SentimentDictionaryBinary or SentimentDictionaryWeighted

Note

Entries in SentimentDictionaryWeighted with a weight of 0 are not counted here

See Also

numEntries numNegativeEntries

Examples

```
numPositiveEntries(SentimentDictionary(c("increase", "rise", "more"),
                           c("fall", "drop"))) # returns 3
numPositiveEntries(SentimentDictionary(c("increase", "decrease", "exit"),
                             c(+1, -1, -10),
                             rep(NA, 3))) # returns 1
```

---

preprocessCorpus *Default preprocessing of corpus to generate a document-term matrix*

Description

Preprocess existing corpus of type Corpus according to default operations. This helper function groups all standard preprocessing steps such that the usage of the package is more convenient. The result is a document-term matrix.

Usage

```
preprocessCorpus(corpus, language = "english", stemming = TRUE,
  verbose = FALSE)
```

Arguments

corpus	Corpus object which should be processed
language	default language used for preprocessing (i.e. stop word removal and stemming)
stemming	perform stemming (default: TRUE)
verbose	print preprocessing status information

See Also

DocumentTermMatrix TermDocumentMatrix

---

print.SentimentDictionaryWordlist *Output content of sentiment dictionary*

Description

Prints entries of sentiment dictionary to the secreen

Usage

```
## S3 method for class 'SentimentDictionaryWordlist'
print(x, ...)

## S3 method for class 'SentimentDictionaryBinary'
print(x, ...)

## S3 method for class 'SentimentDictionaryWeighted'
print(x, ...)
```

Arguments

x	Sentiment dictionary of type SentimentDictionaryWordlist, SentimentDictionaryBinary or SentimentDictionaryWeighted
...	Additional parameters passed to specific sub-routines

See Also

summary for showing a brief summary

Examples

```
print(SentimentDictionary(c("uncertain", "possible", "likely")))
print(SentimentDictionary(c("increase", "rise", "more"),
                          c("fall", "drop")))
print(SentimentDictionary(c("increase", "decrease", "exit"),
                          c(+1, -1, -10),
                          rep(NA, 3)))
```

---

read *Read dictionary from text file*

Description

This routine reads a sentiment dictionary from a text file. Such a text file can be created e.g. via write. The dictionary type is recognized according to the internal format of the file.

Usage

```
read(file)
```

Arguments

file	File name pointing to text file

Value

Dictionary of type SentimentDictionaryWordlist, SentimentDictionaryBinary or SentimentDictionaryWeighte

See Also

write for creating such a file

Examples

```
d.out <- SentimentDictionary(c("uncertain", "possible", "likely"))
write(d.out, "example.dict")
d.in <- read("example.dict")
print(d.in)

d.out <- SentimentDictionary(c("increase", "rise", "more"),
                             c("fall", "drop"))
write(d.out, "example.dict")
d.in <- read("example.dict")
print(d.in)

d.out <- SentimentDictionary(c("increase", "decrease", "exit"),
                             c(+1, -1, -10),
                             rep(NA, 3),
                             intercept=5)
write(d.out, "example.dict")
d.in <- read("example.dict")
print(d.in)

unlink("example.dict")
```

SentimentAnalysis *SentimentAnalysis: A package for analyzing sentiment of texts*

Description

The SentimentAnalysis package provides routines to quickly measure the sentiment of written materials. It ships a dedicated class SentimentDictionary to store different variants of dictionaries (including pre-built ones that are ready to go) and helps the user with routines for constructing domain-specific dictionaries and evaluating the performance of common rules for analyzing sentiment.

SentimentDictionary *Create new sentiment dictionary based on input*

Description

Depending on the input, this function creates a new sentiment dictionary of different type.

Usage

```
SentimentDictionary(...)
```

Arguments

...	Arguments as passed to one of the three functions SentimentDictionaryWordlist, SentimentDictionaryBinary or SentimentDictionaryWeighted

See Also

SentimentDictionaryWordlist SentimentDictionaryBinary SentimentDictionaryWeighted

SentimentDictionaryBinary

Create a sentiment dictionary of positive and negative words

Description

This routines creates a new object of type `SentimentDictionaryBinary` that stores two separate vectors of negative and positive words

Usage

```
SentimentDictionaryBinary(positiveWords, negativeWords)
```

Arguments

`positiveWords`	is a vector containing the entries labeled as positive
`negativeWords`	is a vector containing the entries labeled as negative

Value

Returns a new object of type `SentimentDictionaryBinary`

See Also

`SentimentDictionary`

Examples

```
# generate a dictionary with positive and negative words
d <- SentimentDictionaryBinary(c("increase", "rise", "more"),
                               c("fall", "drop"))
summary(d)
# alternative call
d <- SentimentDictionary(c("increase", "rise", "more"),
                         c("fall", "drop"))
summary(d)
```

SentimentDictionaryWeighted

Create a sentiment dictionary of words linked to a score

Description

This routine creates a new object of type `SentimentDictionaryWeighted` that contains a number of words, each linked to a continuous score (i.e. weight) for specifying its polarity. The scores can later be interpreted as a linear model

Usage

```
SentimentDictionaryWeighted(words, scores, idf, intercept = 0)
```

Arguments

words	is collection (vector) of different words as strings
scores	are the corresponding socres or weights denoting the word's polarity
idf	provide further details on the frequency of words in the corpus as an additional source for normalization
intercept	is an optional parameter for shifting the zero level (default: 0)

Value

Returns a new object of type SentimentDictionaryWordlist

Note

The intercept is useful when the mean or median of a response variable is not exactly located at zero. For instance, stock market returns have slight positive bias.

Source

http://dx.doi.org/10.2139/ssrn.2522884

References

Pr\"ollochs and Feuerriegel (2015). Generating Domain-Specific Dictionaries Using Bayesian Learning. 23rd European Conference on Information Systems (ECIS 2015).

See Also

SentimentDictionary

Examples

```
# generate dictionary (based on linear model)
d <- SentimentDictionaryWeighted(c("increase", "decrease", "exit"),
                                 c(+1, -1, -10),
                                 rep(NA, 3))
summary(d)
# alternative call
d <- SentimentDictionary(c("increase", "decrease", "exit"),
                         c(+1, -1, -10),
                         rep(NA, 3))
summary(d)
```

SentimentDictionaryWordlist

Create a sentiment dictionary consisting of a simple wordlist

---

Description

This routine creates a new object of type SentimentDictionaryWordlist

Usage

```
SentimentDictionaryWordlist(wordlist)
```

Arguments

wordlist	is a vector containing the individual entries as strings

Value

Returns a new object of type SentimentDictionaryWordlist

See Also

SentimentDictionary

Examples

```
# generate a dictionary with "uncertainty" words
d <- SentimentDictionaryWordlist(c("uncertain", "possible", "likely"))
summary(d)
# alternative call
d <- SentimentDictionary(c("uncertain", "possible", "likely"))
summary(d)
```

---

summary.SentimentDictionaryWordlist

Output summary information on sentiment dictionary

Description

Output summary information on sentiment dictionary

Usage

```
## S3 method for class 'SentimentDictionaryWordlist'
summary(object, ...)

## S3 method for class 'SentimentDictionaryBinary'
summary(object, ...)

## S3 method for class 'SentimentDictionaryWeighted'
summary(object, ...)
```

Arguments

object	Sentiment dictionary of type SentimentDictionaryWordlist, SentimentDictionaryBinary or SentimentDictionaryWeighted
...	Additional parameters passed to specific sub-routines

See Also

print for output the entries of a dictionary

Examples

```
summary(SentimentDictionary(c("uncertain", "possible", "likely")))
summary(SentimentDictionary(c("increase", "rise", "more"),
                            c("fall", "drop")))
summary(SentimentDictionary(c("increase", "decrease", "exit"),
                            c(+1, -1, -10),
                            rep(NA, 3)))
```

---

transformIntoCorpus *Transforms the input into a Corpus object*

Description

Takes the given input of characters and transforms it into a Corpus. The input is checked to match the expected class and format.

Usage

```
transformIntoCorpus(x)
```

Arguments

x	A list, data.frame or vector consisting of characters

Value

The generated Corpus

Note

Factors are automatically casted into characters but with printing a warning

See Also

preprocessCorpus for further preprocessing, analyzeSentiment for subsequent sentiment analysis

Examples

```
transformIntoCorpus(c("Document 1", "Document 2", "Document 3"))
transformIntoCorpus(list("Document 1", "Document 2", "Document 3"))
transformIntoCorpus(data.frame("Document 1", "Document 2", "Document 3"))
```

write *Write dictionary to text file*

Description

This routine exports a sentiment dictionary to a text file which can be the source for additional problems or controlling the output.

Usage

```
write(d, file)

## S3 method for class 'SentimentDictionaryWordlist'
write(d, file)

## S3 method for class 'SentimentDictionaryBinary'
write(d, file)

## S3 method for class 'SentimentDictionaryWeighted'
write(d, file)
```

Arguments

d	Dictionary of type SentimentDictionaryWordlist, SentimentDictionaryBinary or SentimentDictionaryWeighted
file	File to which the dictionary should be exported

See Also

read for later access

Examples

```
d.out <- SentimentDictionary(c("uncertain", "possible", "likely"))
write(d.out, "example.dict")
d.in <- read("example.dict")
print(d.in)

d.out <- SentimentDictionary(c("increase", "rise", "more"),
                             c("fall", "drop"))
write(d.out, "example.dict")
d.in <- read("example.dict")
print(d.in)

d.out <- SentimentDictionary(c("increase", "decrease", "exit"),
                             c(+1, -1, -10),
                             rep(NA, 3),
                             intercept=5)
write(d.out, "example.dict")
d.in <- read("example.dict")
print(d.in)

unlink("example.dict")
```

Index