Package: sentixr (via r-universe)

June 24, 2026

Title Lexicons and Tools for Italian Sentiment Analysis

Version 0.2.0

Description Lexicons and tools to perform sentiment analysis on Italian texts. Lexicons included: Sentix 3.0, MAL, ElIta VAD and basic emotions (Plutchik's wheel of emotions). For more details about the lexicons, see Basile & Nissim (2013), ``Sentiment Analysis on Italian Tweets'', <https://aclanthology.org/W13-1614/>; Vassallo et al. (2019), ``The Tenuousness of Lemmatization in Lexicon-based Sentiment Analysis'', <https://aclanthology.org/2019.clicit-1.79/>; Di Palma (2024), ``ELIta: A New Italian Language Resource for Emotion Analysis'', <https://aclanthology.org/2024.clicit-1.36/>.

Language en, it

Depends R (>= 4.1), dplyr, tidyselect, rlang

Imports udpipe

Suggests knitr, rmarkdown, tidytext, spacyr, readtext, tibble, testthat (>= 3.0.0), remotes, quanteda, mockery

VignetteBuilder knitr

License GPL (>= 3)

Encoding UTF-8

LazyData true

NeedsCompilation no

URL https://github.com/valeriobasile/sentixr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Author Agnese Vardanega [aut, cre] (ORCID: <https://orcid.org/0000-0002-1419-9896>), Valerio Basile [aut] (ORCID: <https://orcid.org/0000-0001-8110-6832>), Eliana Di Palma [aut] (ORCID: <https://orcid.org/0000-0003-2154-2696>), Giuliano Gabrieli [aut] (ORCID:

<https://orcid.org/0009-0005-1153-5662>), Marco Vassallo [aut]
(ORCID: <https://orcid.org/0000-0001-7016-6549>)

Maintainer Agnese Vardanega <avardanega@unite.it>

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

Date/Publication 2026-06-24 08:37:41 UTC

RemoteUrl https://github.com/cran/sentixr

RemoteRef HEAD

RemoteSha cd9dd4245343d49218f2c4ba154b274f8d78a38a

Contents

df_to_dict 2
df_to_polar 4
df_to_valence 5
elita_basic 6
elita_VAD 7
get_elita 8
get_sentix 9
make_polarity 10
MAL 11
recensioni_tv 12
sentix 13
sentix_annotate 14
sentix_summarize 16

Index **19**

`df_to_dict` *Convert a data frame to a Quanteda dictionary with polarity or valence*

Description

Converts a data frame (tibble) containing a lexicon into a Quanteda dictionary with valence or polarity. Requires the package *Quanteda*. If the `quanteda.sentiment` package is also installed, the polarity or valence attributes will be detected and assigned automatically. Otherwise, a standard Quanteda dictionary will be created.

The function is a wrapper for `df_to_valence()` and `df_to_polar()`, automatically determining, where possible, the most appropriate type of dictionary for the input data frame (see Details).

Note: The function cannot handle duplicate entries, and will remove rows with NAs.

Usage

```
df_to_dict(
  x,
  word_field = NULL,
  type = "auto",
  polar_field = "polarity",
  polar_map = NULL
)
```

Arguments

x	A data.frame or tibble with at least one character column with the terms, and either numeric columns (scores) or a categorical column (with polarity).
word_field	A string with the name of the column containing the terms. If NULL (the default), the function will try to detect a column named "lemma" or "word". If neither is found, it selects the first character column available.
type	The type of dictionary to create. Can be "auto" (the default), "valence", or "polarity".
polar_field	A string with the name of the column containing the categories (polarities; i.e. "Positive", "Negative"). Defaults to "polarity". Ignored for valence dictionaries.
polar_map	A named character vector to manually map dictionary keys to standard polarity values (pos, neg, neut). Example: c(pos = "Positive", neg = "Negative"). If NULL (default), the function will attempt to auto-detect the mapping.

Details

The function handles the sentiment scores or categories as follows:

- **Valence Dictionaries**: The names of the numeric columns are used as dictionary keys. When there is only one numeric column, the word_field is used as the key name (see quanteda.sentiment::valence if installed).
- **Polarity Dictionaries**:
  - The character or factor column (other than the word_field) is used to group terms into the categories (polar_field) that are then associated with the standard "polarity" attribute ("pos", "neg", optionally "neut"; see quanteda.sentiment::polarity if installed).
  - The "polarity" attribute is assigned via the polar_map argument, or automatically if the categories in the polar_field are explicit: "positive", "negative" (and, optionally, "neutral"; case-insensitive).

Value

A quanteda::dictionary2 object.

See Also

df_to_dict(), df_to_polar(), dictionary

Examples

```
if(requireNamespace("quanteda")){
# only numeric fields are present
my_dict <- get_sentix()
df_to_dict(my_dict)

# no numeric fields are present
my_dict <- get_sentix(polarity = TRUE)
df_to_dict(my_dict)
}
```

`df_to_polar` *Convert a data frame to a Quanteda polarity dictionary*

Description

Converts a data frame (tibble) containing a lexicon into a Quanteda dictionary with polarity, to be used with `quanteda.sentiment::textstat_polarity()`. Requires the package *Quanteda*. If the `quanteda.sentiment` package is also installed, the polarity attribute will be detected and assigned automatically. Otherwise, a standard Quanteda dictionary will be created.

Note: The function cannot handle duplicate entries, and will remove rows with NAs.

Usage

```
df_to_polar(x, word_field = NULL, polar_field = "polarity", polar_map = NULL)
```

Arguments

`x`	A `data.frame` or `tibble` with at least one character column with the terms, and either numeric columns (scores) or a categorical column (with polarity).
`word_field`	A string with the name of the column containing the terms. If `NULL` (the default), the function will try to detect a column named "lemma" or "word". If neither is found, it selects the first character column available.
`polar_field`	A string with the name of the column containing the categories (polarities; i.e. "Positive", "Negative"). Defaults to `"polarity"`.
`polar_map`	A named character vector to manually map dictionary keys to standard polarity values (`pos`, `neg`, `neut`). Example: `c(pos = "Positive", neg = "Negative")`. If `NULL` (default), the function will attempt to auto-detect the mapping.

Details

The function handles the sentiment categories as follows:

- The character or factor column (other than the `word_field`) is used to group terms into the categories (`polar_field`) that are then associated with the standard "polarity" attribute ("`pos`", "`neg`", optionally "`neut`"; see `quanteda.sentiment::polarity` if installed).
- The "polarity" attribute is assigned via the `polar_map` argument, or automatically if the categories in the `polar_field` are explicit: "positive", "negative" (and, optionally, "neutral"; case-insensitive).

Value

A quanteda::dictionary2 object.

See Also

df_to_dict(), df_to_valence(), dictionary

Examples

```
if(requireNamespace("quanteda")){
# Create a polarity dictionary from sentix
my_dict <- get_sentix(polarity = TRUE)
my_pol_dict <- df_to_polar(my_dict)
}
```

---

df_to_valence *Convert a data frame to a Quanteda valence dictionary*

Description

Converts a data frame (tibble) containing a lexicon into a Quanteda dictionary with valence, to be used with quanteda.sentiment::textstat_valence(). Requires the package *Quanteda*. If the quanteda.sentiment package is also installed, the valence attribute will be detected and assigned automatically. Otherwise, a standard Quanteda dictionary will be created.

Note: The function cannot handle duplicate entries, and will remove rows with NAs.

Usage

```
df_to_valence(x, word_field = NULL)
```

Arguments

x	A data.frame or tibble with at least one character column with the terms, and either numeric columns (scores) or a categorical column (with polarity).
word_field	A string with the name of the column containing the terms. If NULL (the default), the function will try to detect a column named "lemma" or "word". If neither is found, it selects the first character column available.

Details

The names of the numeric columns are used as dictionary keys. When there is only one numeric column, the word_field is used as the key name (see quanteda.sentiment::valence if installed).

Value

A quanteda::dictionary2 object.

See Also

df_to_dict(), df_to_polar(), dictionary

Examples

```
if(requireNamespace("quanteda")){
# Create a valence dictionary from elita_VAD
data(elita_VAD)
elita_dict <- df_to_valence(elita_VAD)
}
```

---

elita_basic *ELIta: Basic Emotions*

Description

A dataset containing scores for 6,905 Italian lexical entries (lemmas and emojis) on the eight basic emotions of Plutchik's wheel together with the dyad *love*, formed by the combination of *trust* and *joy* (Plutchik 1980). It uses a scale from "non associated" (0), "weakly associated" (0.25), "moderately associated" (0.75) to "strongly associated" (1).

This dataset is a subset of the broader ELIta framework: see elita_VAD, for the VAD dimensional approach (Valence, Arousal, and Dominance)

Usage

```
data(elita_basic)
```

Format

A tibble with 6,905 rows and 10 columns:

lemma Italian lemmas and emojis (character).

gioia Joy: 0, +1, (double).

tristezza Sadness: 0, +1, (double).

rabbia Anger: 0, +1, (double).

disgusto Disgust: 0, +1, (double).

paura Fear: 0, +1, (double).

fiducia Trust: 0, +1, (double).

sorpresa Surprise: 0, +1, (double).

aspettativa Anticipation: 0, +1, (double).

amore Love: 0, +1, (double).

Note

Source

GitHub Repository: https://github.com/elianadipalma/ELIta

References

Di Palma, E. (2024a). ELIta: A New Italian Language Resource for Emotion Analysis. *Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024)*, 297–307. https://aclanthology.org/2024.clicit-1.36/

Di Palma, E. (2024b). ELIta (Emotion Lexicon for Italian). http://hdl.handle.net/20.500.11752/OPEN-1036

Plutchik, R. (1980). A general psychoevolutionary theory of emotion. In R. Plutchik & H. Kellerman (eds.), *Theories of Emotion* (pp. 3–33). Academic Press.

See Also

elita_VAD, get_elita().

Examples

```
data(elita_basic)
get_elita(dict = "elita_basic")
```

---

elita_VAD *ELIta: VAD Dimensions (Valence, Arousal, Dominance)*

Description

A dataset containing scores for 6,905 Italian lexical entries (lemmas and emojis) on the VAD dimensions (Valence, Arousal, and Dominance; see Russell 1980)

This dataset is a subset of the broader ELIta framework. See elita_basic, for basic discrete emotions (Plutchik's wheel).

Usage

```
data(elita_VAD)
```

Format

A tibble with 6,905 rows and 4 columns:

lemma Italian lemmas and emojis (character).

valenza Valence (unpleasant - pleasant): -4, +4 (double).

attivazione Arousal (calm - excited/active): -4, +4 (double).

dominanza Dominance (submissive/controlled - dominant/in control): -4 to +4 (double).

Note

The dataset is distributed under the Creative Commons Universal License (CC0 1.0).

Source

GitHub Repository: https://github.com/elianadipalma/ELIta

References

Di Palma, E. (2024a). ELIta: A New Italian Language Resource for Emotion Analysis. *Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024)*, 297–307. https://aclanthology.org/2024.clicit-1.36/

Di Palma, E. (2024b). ELIta (Emotion Lexicon for Italian). http://hdl.handle.net/20.500.11752/OPEN-1036

Russell, J. A. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6), 1161–1178.

See Also

elita_basic, get_elita().

Examples

```
data(elita_VAD)

# To rescale scores to -1, + 1
get_elita(dict = "elita_VAD")
```

---

get_elita *Get ELIta-family Lexicons*

Description

A utility function to access ELIta-family lexicons (elita_basic, elita_VAD) with convenient defaults.

For all lexicons, it returns the key entry (`lemma` or `word`) and `score` columns, suitable for joining.

For `elita_VAD`, which contains scores on a -4 to +4 scale, scores are "centered" by default (divided by 4 to map to a -1 to +1 theoretical range).

Usage

```
get_elita(dict = "elita_VAD", rescale = "default")
```

Arguments

dict	The name of the lexicon to retrieve. Must be one of: "elita_VAD" (default) or "elita_basic".
rescale	Character string indicating the rescaling method applied to the scores. Options are: • "default": For elita_VAD = "centered". For elita_basic = "none". • "none": No rescaling is applied. • "centered": Scores are divided by their theoretical maximum (4 for elita_VAD) to map to -1/+1. • "normalized": Scores are normalized empirically to -1/+1 (score / max(abs(.x), na.rm = TRUE)). This argument only applies elita_VAD. elita_basic scores are within 0,1,

Value

A tibble.

See Also

elita_VAD, elita_basic, get_sentix()

Examples

```
# Get the default elita_VAD lexicon (centered scores)
my_dict_VAD <- get_elita("elita_VAD")

# Get elita_VAD without any rescaling
my_dict_VAD <- get_elita("elita_VAD", rescale = "none")

# Get elita_basic lexicon
my_dict_basic <- get_elita("elita_basic")
```

---

get_sentix *Get Sentix-family Lexicons*

Description

A utility function to access Sentix-family lexicons (sentix, MAL) with convenient defaults.

For all lexicons, it returns by default the key entry (lemma or word) and score columns, suitable for joining. Polarity classification can be computed via make_polarity().

Other columns (polypathy_index) are accessible via arguments.

Usage

```
get_sentix(
  dict = "sentix",
  polypathy = FALSE,
  polarity = FALSE,
  polar_field = "polarity",
  threshold = 0
)
```

Arguments

dict	The name of the lexicon to retrieve. Must be one of: "sentix" (default) or "MAL".
polypathy	Logical. If TRUE, the polypathy_index column is included. Defaults to FALSE.
polarity	Logical. If TRUE, a polarity column is added (computed via make_polarity). Defaults to FALSE.
polar_field	Character string. The name of the new polarity column. Defaults to "polarity".
threshold	Numeric. The threshold for make_polarity (positive value). Scores within [-threshold, threshold] are considered neutral. Defaults to 0.

Value

A tibble.

See Also

sentix, MAL, get_elita(), make_polarity()

Examples

```
# Get the default sentix lexicon (key and score)
my_dict <- get_sentix()

# Get the sentix lexicon with polarity field
my_dict <- get_sentix(polarity = TRUE)

# Get MAL and polypathy index
my_dict_poly <- get_sentix("MAL", polypathy = TRUE)
```

---

make_polarity *Classify sentiment scores into polarity categories*

Description

Utility function for adding polarity columns to sentiment lexicons, to be used within mutate.

Classifies numeric sentiment scores into "positive", "negative", or "neutral", based on a specified threshold (defaults to 0).

Usage

```
make_polarity(score, threshold = 0)
```

Arguments

score	A numeric vector of sentiment scores.
threshold	A numeric vector. If length 1 (i.e. `threshold = 0.125`), it must be a positive value, that will be used as absolute threshold; if length 2 (i.e. `threshold = c(0.125, -0.135)`), it must contain one positive (or 0) and one negative (or 0) value. Defaults to 0. Scores `>= threshold` and `<= -threshold` will be classified as "positive" and "negative", respectively. 0 is always classified as "neutral".

Value

A character vector with "positive", "negative", or "neutral".

Examples

```
sentix |>
  mutate(polarity = make_polarity(score))

# with custom threshold
elita_VAD |>
  mutate(across(where(is.numeric),
                      ~ make_polarity(.x, 0.125)))

# with custom asymmetric thresholds
get_sentix("MAL") |>
 mutate(polarity = make_polarity(score,
                                 threshold = c(0.125, -0.135)))
```

MAL *MAL 3.1. Affective Lexicon*

Description

MAL (Morphologically-inflected Affective Lexicon) is an affective lexicon for the Italian language. It expands `sentix` with inflected forms from *Morph-it!* (Vassallo et al. 2019; see Zanchetta & Baroni 2005), and can be therefore used without lemmatization.

It contains 295,032 inflected forms (field `word`), with associated affective scores, and an index of polypathy.

Affective scores are inherited from the corresponding `sentix` entries (lemmas).

Usage

```
data(MAL)
```

Format

A tibble with 297,592 rows and 4 columns:

lemma Italian inflected forms (character).

score Sentiment valence: -1, +1 (double).

polypathy_index Index of ambiguity: `"0"`, `"1"`, `"2"`, `"3"` (ordered factor; see `sentix` for details).

Note

The dataset is distributed under the CC BY-SA 4.0 license.

Source

Zenodo Repository: https://zenodo.org/records/18709688.

References

Vassallo, M., Gabrieli, G., Basile, V., & Bosco, C. (2019). The Tenuousness of Lemmatization in Lexicon-based Sentiment Analysis. In *Proceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019)*, pages 520–525, Bari, Italy. CEUR Workshop Proceedings. https://aclanthology.org/2019.clicit-1.79/

Zanchetta, E., & Baroni, M. (2005). Morph-it! A free corpus-based morphological resource for the italian language. In *Proceedings of Corpus linguistics Conference Series 2005*, University of Birmingham. https://cris.unibo.it/handle/11585/15321

See Also

`sentix`, `get_sentix()`

Examples

```
data(MAL)
get_sentix(dict = "MAL")
```

---

recensioni_tv *Example text data for sentiment analysis*

Description

A dataset containing 5 sentences in Italian, derived from TV reviews on Amazon, for testing and demonstrating the package functions.

Usage

```
data(recensioni_tv)
```

Format

A tibble with 5 rows and 2 variables:

doc_id Unique identifier for the document (doc1 to doc5)

text The text content of the reviews

`sentix` *Sentix 3.1. Affective Lexicon*

Description

Sentix is an affective lexicon for the Italian language (Basile & Nissim 2013; Basile et al. 2025).

It includes 68,190 italian lemmas (field `lemma`) with associated affective scores and an index of polypathy (see: Details).

Usage

```
data(sentix)
```

Format

A tibble with 68,190 rows and 4 columns:

lemma Italian lemmas (character).

score Sentiment valence: -1, +1 (double).

polypathy_index Index of ambiguity (see Details): `"0"`, `"1"`, `"2"`, `"3"` (ordered factor).

Details

The `polypathy_index` provides information on the ambivalence and stability of the sentiment scores, on the basis of the original multiple entries for each lemma. The values are interpreted as follows:

- `"0"`: No multiple entries for the lemma.
- `"1"`: Multiple entries with a low range (`max - min`) of original scores.
- `"2"`: Multiple entries with a high range of original scores.
- `"3"`: Multiple entries with a high range of original scores, and ambivalence (sign change).

Note

The dataset is distributed under the CC BY-SA 4.0 license.

Source

- Zenodo Repository: https://zenodo.org/records/15609186.
- GitHub Repository: https://github.com/valeriobasile/sentix

References

Basile, V., & Nissim, M. (2013). Sentiment Analysis on Italian Tweets. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 100–107, Atlanta, Georgia. Association for Computational Linguistics. https://aclanthology.org/W13-1614/.

Basile, V., Nissim, M., Bosco, C., Vassallo, M., & Gabrieli, G. (2025). *Sentix* (3.1). Zenodo. doi:10.5281/zenodo.15609185.

See Also

MAL, get_sentix()

Examples

```
data(sentix)
get_sentix()
```

`sentix_annotate` *Annotate text with sentiment scores*

Description

Annotates a character vector or a data frame of texts using udpipe (providing a model where necessary) and joins the results with a selected sentiment lexicon.

Usage

```
sentix_annotate(
  x,
  model = NULL,
  text_field = NULL,
  docid_field = NULL,
  dict = "sentix",
  rescale = "default",
  simplify = TRUE,
  ...
)
```

Arguments

`x`	A character vector, a data frame, or a list of texts.
`model`	A UDpipe model used for annotation. The following options are supported: • `NULL` (default): The function will automatically download and load the default Italian model (`italian-isdt`). Requires an internet connection on first use.

- `"local"`: Searches for a single `.udpipe` model file in the current working directory.
- A character string providing the full path to a `.udpipe` file ("path/to/model").
- A model object already loaded with `udpipe_load_model`. This is the most efficient option for repeated calls.

`text_field` A character string specifying the name of the column containing the text to be parsed. If `NULL` (the default), the function will try to detect column named "text".

`docid_field` A character string specifying the name of the column containing document identifiers. If `NULL` (the default), the function will try to detect a column named "doc_id"; if not found, IDs will be automatically generated.

`dict` The name of the lexicon to use. Can be one of the Sentix family (`"sentix"`, `"MAL"`) or ELIta family (`"elita_VAD"`, `"elita_basic"`). Defaults to `"sentix"`.

`rescale` Character string indicating the rescaling method for scores: `"default"` (for the lexicon), `"none"`, `"centered"`, `"normalized"`. Defaults to `"default"`. Passed to `get_elita()`.

`simplify` Logical. Defaults to `TRUE`. If `FALSE`, returns standard UDpipe columns (see Details).

`...` Additional arguments passed to the lexicon retrieval function (see `get_sentix()`, `get_elita()`). OR to `udpipe` and subsequently to `udpipe_annotate`. Note: `tagger = "none"` is not supported (it will be ignored), as POS tagging is required for lemmatization and punctuation filtering.

Details

This function uses `udpipe` to process the input texts, then joins the tokenized output with the specified lexicon, using *dplyr* `join`. It performs two main steps:

1. **Parsing:** Texts are tokenized, tagged, and lemmatized. For larger corpora, the function supports parallel processing by passing the argument `parallel.cores` to `udpipe`.
2. **Matching:** The parsed tokens are joined with the selected lexicon using `left_join`.

All tokens are thus preserved in the output to maintain context, with `NA` assigned to tokens not found in the lexicon.

Value

A `tibble` with one row per token, containing the following columns: `doc_id`, `sentence_id`, `token_id`, `token`, `lemma`, `upos`, one or more sentiment score columns, named after those of the selected lexicon.

When `simplify = FALSE` it will include standard UDpipe columns (see `as.data.frame.udpipe_connlu`), plus sentiment score columns.

See Also

`get_sentix()`, `get_elita()`, `sentix_summarize()`

Examples

```
## Not run:
# This example is not executed because it requires the udpipe package and
# downloading a model
# Auto-download model
ann_df <- sentix_annotate("Oggi è una bella giornata")

# Use a local model file in the working directory (i.e. if already
# downloaded)
ann_df <- sentix_annotate("Uso un modello locale.", model = "local")

# Use specific model path and lexicon
ann_df <- sentix_annotate("Oggi è una bella giornata",
    model = "path/to/model.udpipe",
    dict = "elita_VAD")

# With a data frame, and a loaded model
data("recensioni_tv")
model <- udpipe::udpipe_load_model("italian-isdt-ud-2.5-191206.udpipe")
ann_df <- sentix_annotate(recensioni_tv, model = model)

## End(Not run)
```

---

sentix_summarize *Summarize sentiment annotations*

Description

Calculates sentiment scores and, optionally, ambiguity metrics, aggregating token-level sentiment annotations to the document level.

Usage

```
sentix_summarize(
  x,
  aggregation = "mean",
  cols = NULL,
  by = "doc_id",
  simplify = FALSE,
  ambiguity = "3"
)
```

Arguments

x	A data frame containing at least a `doc_id` column and numeric columns with sentiment scores.
aggregation	Character. `"mean"` (default) or `"sum"`.

cols	Character vector, specifying columns to summarize. If NULL (default), numeric columns will be automatically considered as scores.
by	Character vector, specifying the column(s) to group by. Defaults to doc_id.
simplify	Logical. Defaults to FALSE. When TRUE, the output will contain only basic measures (score and, where applicable ambiguity).
ambiguity	Character. The minimum polypathy_index value to be considered. Can be "1", "2", "3" (default), or "none" to disable ambiguity calculation (see Details). Ignored if the column polypathy_index is not present in the input.

Details

This function takes the output of sentix_annotate() or a data frame or tibble withat least a doc_id column and sentiment scores (numeric columns).

Metrics Calculated:

- score: the average (or sum) of the sentiment columns.
- ambiguity: n_poly / n_scored (if polypathy_index is present).
- n_tokens: total valid tokens, excluding punctuation. UDpipe's CoNLL-U format expands Multi-Word Tokens (MWTs) into their syntactic components, including articulated prepositions: e.g., 'nella' becomes 'in' + 'la'. The count only considers the components (e.g., 'nella' counts for 2 tokens, not 3).
- n_scored: tokens with *at least one* sentiment score.
- n_poly: count of ambiguous tokens, based on the ambiguity level setting, and if the column polypathy_index is present in the lexicon.

Value

A tibble with one row per document.

See Also

get_sentix(), sentix, sentix_annotate()

Examples

```
## Not run:
# This example is not executed because it requires the udpipe package and
# downloading a model
testo <- "Oggi è una bella giornata. Uscirò a fare una passeggiata"
# With the output of sentix_annotate
ann_df <- sentix_annotate(testo, model = "local")
sentix_summarize(ann_df)
# With only basic measures
sentix_summarize(ann_df, simplify = TRUE)
# With custom grouping (e.g., per sentence)
sentix_summarize(ann_df, by =c("doc_id", "sentence_id"))
# With the output of sentix_annotate, ambiguity and other intermediate
# measures
```

```
ann_df <- sentix_annotate(testo,
                          polypathy = TRUE,
                          model = "local")
sentix_summarize(ann_df)

## End(Not run)
```

Index

∗ **datasets**
    recensioni_tv, 12
∗ **functions**
    df_to_dict, 2
    df_to_polar, 4
    df_to_valence, 5
    get_elita, 8
    get_sentix, 9
    make_polarity, 10
    sentix_annotate, 14
    sentix_summarize, 16
∗ **lexicon**
    elita_basic, 6
    elita_VAD, 7
    MAL, 11
    sentix, 13

as.data.frame.udpipe_connlu, *15*

df_to_dict, 2
df_to_dict(), *3*, *5*, *6*
df_to_polar, 4
df_to_polar(), *2*, *3*, *6*
df_to_valence, 5
df_to_valence(), *2*, *5*
dictionary, *3*, *5*, *6*

elita_basic, 6, *7–9*
elita_VAD, *6*, 7, *7–9*

get_elita, 8
get_elita(), *7*, *8*, *10*, *15*
get_sentix, 9
get_sentix(), *9*, *12*, *14*, *15*, *17*

join, *15*

left_join, *15*

make_polarity, 10, *10*
make_polarity(), *9*, *10*
MAL, *9*, *10*, 11, *14*
mutate, *10*

recensioni_tv, 12

sentix, *9–12*, 13, *17*
sentix_annotate, 14
sentix_annotate(), *17*
sentix_summarize, 16
sentix_summarize(), *15*

udpipe, *14*, *15*
udpipe_annotate, *15*
udpipe_load_model, *15*