Package ‘transforEmotion’

May 11, 2022

Title Sentiment Analysis for Text and Qualitative Data

Version 0.1.1

Date 2022-05-11

Maintainer Alexander Christensen <alexpaulchristensen@gmail.com>

Description Implements sentiment analysis using huggingface <https://huggingface.co> transformer zero-shot classification model pipelines. The default pipeline is Cross-Encoder's DistilRoBERTa <https://huggingface.co/cross-encoder/nli-distilroberta-base> trained on the Stanford Natural Language Inference <https://nlp.stanford.edu/projects/snli/> and Multi-Genre Natural Language Inference <https://huggingface.co/datasets/multi_nli> datasets. Using similar models, zero-shot classification transformers have demonstrated superior performance relative to other natural language processing models <arXiv:1909.00161>. All other zero-shot classification model pipelines can be implemented using their model name from <https://huggingface.co/models?pipeline_tag=zero-shot-classification>}.

Depends R (>= 3.5.0)

License GPL (>= 3.0)

Encoding UTF-8

Imports reticulate, pbapply, osfr, LSAfun, dplyr, remotes

Suggests markdown, knitr, rmarkdown, rstudioapi

VignetteBuilder knitr

RoxygenNote 7.1.2

NeedsCompilation no

Author Alexander Christensen [aut, cre] (<https://orcid.org/0000-0002-9798-7037>), Hudson Golino [aut] (<https://orcid.org/0000-0002-1601-1447>)

Repository CRAN

Date/Publication 2022-05-11 12:30:02 UTC

R topics documented:

`transforEmotion-package` *transforEmotion–package*

Description

Implements sentiment analysis using huggingface transformer zero-shot classification model pipelines. The default pipeline is Cross-Encoder's DistilRoBERTa trained on the Stanford Natural Language Inference (SNLI) and Multi-Genre Natural Language Inference (MultiNLI) datasets. Using similar models, zero-shot classification transformers have demonstrated superior performance relative to other natural language processing models (Yin, Hay, & Roth, 2019). All other zero-shot classification model pipelines can be implemented using their model name from https://huggingface.co/models?pipeline_tag=zero-shot-classification.

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com> and Hudson Golino <hfg9s@virginia.edu>

References

Yin, W., Hay, J., & Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. arXiv preprint arXiv:1909.00161.

`emotions` *Emotions Data*

Description

A matrix containing words (n = 175,592) and the emotion category most frequently associated with each word. This dataset is a modified version of the 'DepecheMood++' lexicon developed by Araque, Gatti, Staiano, and Guerini (2018). For proper scoring, text should not be stemmed prior to using this lexicon. This version of the lexicon does not rely on part of speech tagging.

Usage

```
data(emotions)
```

Format

A data frame with 175,592 rows and 9 columns.

word An entry in the lexicon, in English

AFRAID, AMUSED, ANGRY, ANNOYED, DONT_CARE, HAPPY, INSPIRED, SAD The emotional category. All emotions contain either a 0 or 1. If the category is most likely to be associated with the word, it recieves a 1, otherwise, 0. Words are only associated with one category.

References

Araque, O., Gatti, L., Staiano, J., and Guerini, M. (2018). DepecheMood++: A bilingual emotion lexicon built through simple yet powerful techniques. *ArXiv*

Examples

```
data("emotions")
```

`emoxicon_scores` *Emoxicon Scores*

Description

A bag-of-words approach for computing emotions in text data using the lexicon compiled by Araque, Gatti, Staiano, and Guerini (2018).

Usage

```
emoxicon_scores(text, lexicon, exclude)
```

Arguments

`text`	Matrix or data frame. A data frame containing texts to be scored (one text per row)
`lexicon`	The lexicon used to score the words. The default is the `emotions` dataset, a modification of the lexicon developed by Araque, Gatti, Staiano, and Guerini (2018). To use the raw lexicon from Araque et. al (2018) containing the original probability weights, use the `weights` dataset. If another custom lexicon is used, the first column of the lexicon should contain the terms and the subsequent columns contain the scoring categories.
`exclude`	A vector listing terms that should be excluded from the lexicon. Words specified in `exclude` will not influence document scoring. Users should consider excluding 'red herring' words that are more closely related to the topics of the documents, rather than the documents' emotional content. For example, the words "clinton" and "trump" are present in the lexicon and are both associated with the emotion 'AMUSED'. Excluding these words when analyzing political opinions may produce more accurate results.

Author(s)

Tara Valladares <tls8vx at virginia.edu> and Hudson F. Golino <hfg9s at virginia.edu>

References

Araque, O., Gatti, L., Staiano, J., and Guerini, M. (2018). DepecheMood++: A bilingual emotion lexicon built through simple yet powerful techniques. *ArXiv*

See Also

`emotions`, where we describe how we modified the original DepecheMood++ lexicon.

Examples

```
# Obtain "emotions" data
data("emotions")

# Obtain "tinytrolls" data
data("tinytrolls")

## Not run:
# Obtain emoxicon scores for first 10 tweets
emotions_tinytrolls <- emoxicon_scores(text = tinytrolls$content, lexicon = emotions)

## End(Not run)
```

`neo_ipip_extraversion` *NEO-PI-R IPIP Extraversion Item Descriptions*

Description

A list (length = 6) of the NEO-PI-R IPIP item descriptions (https://ipip.ori.org/newNEOFacetsKey.htm). Each vector within the 6 list elements contains the item descriptions for the respective Extraversion facets – friendliness, gregariousness, assertiveness, activity_level, excitement_seeking, and cheerfulness

Usage

```
data(neo_ipip_extraversion)
```

Format

A list (length = 6)

Examples

```
data("neo_ipip_extraversion")
```

`nlp_scores` *Natural Language Processing Scores*

Description

Natural Language Processing using word embeddings to compute semantic similarities (cosine) of text and specified classes

Usage

```
nlp_scores(
  text,
  classes,
  semantic_space = c("baroni", "cbow", "cbow_ukwac", "en100", "glove", "tasa"),
  preprocess = TRUE,
  remove_stop = TRUE,
  keep_in_env = TRUE,
  envir = 1
)
```

Arguments

`text`	Character vector or list. Text in a vector or list data format
`classes`	Character vector. Classes to score the text
`semantic_space`	Character vector. The semantic space used to compute the distances between words (more than one allowed). Here's a list of the semantic spaces: • `"baroni"` Combination of British National Corpus, ukWaC corpus, and a 2009 Wikipedia dump. Space created using continuous bag of words algorithm using a context window size of 11 words (5 left and right) and 400 dimensions. Best word2vec model according to Baroni, Dinu, & Kruszewski (2014) • `"cbow"` Combination of British National Corpus, ukWaC corpus, and a 2009 Wikipedia dump. Space created using continuous bag of words algorithm with a context window size of 5 (2 left and right) and 300 dimensions • `"cbow_ukwac"` ukWaC corpus with the continuous bag of words algorithm with a context window size of 5 (2 left and right) and 400 dimensions • `"en100"` Combination of British National Corpus, ukWaC corpus, and a 2009 Wikipedia dump. 100,000 most frequent words. Uses moving window model with a size of 5 (2 to the left and right). Positive pointwise mutual information and singular value decomposition was used to reduce the space to 300 dimensions • `"glove"` Wikipedia 2014 dump and Gigaword 5 with 400,000 words (300 dimensions). Uses co-occurrence of words in text documents (uses cosine similarity) • `"tasa"` Latent Semantic Analysis space from TASA corpus all (300 dimensions). Uses co-occurrence of words in text documents (uses cosine similarity)
`preprocess`	Boolean. Should basic preprocessing be applied? Includes making lowercase, keeping only alphanumeric characters, removing escape characters, removing repeated characters, and removing white space. Defaults to `TRUE`
`remove_stop`	Boolean. Should `stop_words` be removed? Defaults to `TRUE`
`keep_in_env`	Boolean. Whether the classifier should be kept in your global environment. Defaults to `TRUE`. By keeping the classifier in your environment, you can skip re-loading the classifier every time you run this function. `TRUE` is recommended
`envir`	Numeric. Environment for the classifier to be saved for repeated use. Defaults to the global environment

Value

Returns semantic distances for the text classes

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd annual meting of the association for computational linguistics* (pp. 238-247).

Landauer, T.K., & Dumais, S.T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, *104*, 211-240.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing* (pp. 1532-1543).

Examples

```
# Load data
data(neo_ipip_extraversion)

# Example text
text <- neo_ipip_extraversion$friendliness[1:5]

## Not run:
# GloVe
nlp_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 )
)

# Baroni
nlp_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 ),
 semantic_space = "baroni"
)

# CBOW
nlp_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 ),
 semantic_space = "cbow"
)

# CBOW + ukWaC
nlp_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 ),
 semantic_space = "cbow_ukwac"
)

# en100
nlp_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 ),
 semantic_space = "en100"
)

# tasa
nlp_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 ),
 semantic_space = "tasa"
)

## End(Not run)
```

`punctuate` *Punctuation Removal for Text*

Description

Keeps the punctuations you want and removes the punctuations you don't

Usage

```
punctuate(
  text,
  allowPunctuations = c("-", "?", "'", "\"", ";", ",", ".", "!")
)
```

Arguments

`text`	Character vector or list. Text in a vector or list data format
`allowPunctuations`	Character vector. Punctuations that should be allowed in the text. Defaults to common punctuations in English text

Details

Coarsely removes punctuations from text. Keeps general punctuations that are used in most English language text. Apostrophes are much trickier. For example, not allowing "'" will remove apostrophes from contractions like "can't" becoming "cant"

Value

Returns text with only the allowed punctuations

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Load data
data(neo_ipip_extraversion)

# Example text
text <- neo_ipip_extraversion$friendliness

# Keep only periods
punctuate(text, allowPunctuations = c("."))
```

`setup_miniconda` *Install Miniconda*

Description

Installs miniconda

Usage

```
setup_miniconda()
```

Details

Installs miniconda using `install_miniconda`

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

`setup_modules` *Install Necessary Python Modules*

Description

Installs modules to compute `transformer_scores`. These include

- pytorch
- torchvison
- torchaudio
- tensorflow
- transformers

Usage

```
setup_modules()
```

Details

Installs modules for miniconda using `conda_install`

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

`stop_words` *Stop Words from the* tm *Package*

Description

174 English stop words in the *tm* package

Usage

```
data(stop_words)
```

Format

A vector (length = 174)

Examples

```
data("stop_words")
```

`tinytrolls` *Russian Trolls Data - Small Version*

Description

A matrix containing a smaller subset of tweets from the `trolls` dataset, useful for test purposes. There are approximately 20,000 tweets from 50 authors. This dataset includes only authored tweets by each account; retweets, reposts, and repeated tweets have been removed. The original data was provided by FiveThirtyEight and Clemson University researchers Darren Linvill and Patrick Warren. For more information, visit https://github.com/fivethirtyeight/russian-troll-tweets

Usage

```
data(tinytrolls)
```

Format

A data frame with 22,143 rows and 6 columns.

content A tweet.

author The name of the handle that authored the tweet.

publish_date The date the tweet was published on.

followers How many followers the handle had at the time of posting.

updates How many interactions (including likes, tweets, retweets) the post garnered.

account_type Left or Right

Examples

```
data(tinytrolls)
```

`transformer_scores` *Sentiment Analysis Scores*

Description

Uses sentiment analysis pipelines from huggingface to compute probabilities that the text corresponds to the specified classes

Usage

```
transformer_scores(
  text,
  classes,
  multiple_classes = FALSE,
  transformer = c("cross-encoder-roberta", "cross-encoder-distilroberta",
    "facebook-bart"),
  preprocess = FALSE,
  keep_in_env = TRUE,
  envir = 1
)
```

Arguments

`text`	Character vector or list. Text in a vector or list data format
`classes`	Character vector. Classes to score the text
`multiple_classes`	Boolean. Whether the text can belong to multiple true classes. Defaults to `FALSE`. Set to `TRUE` to get scores with multiple classes
`transformer`	Character. Specific zero-shot sentiment analysis transformer to be used. Default options: • `"cross-encoder-roberta"` Uses Cross-Encoder's Natural Language Interface RoBERTa Base zero-shot classification model trained on the Stanford Natural Language Inference (SNLI) corpus and MultiNLI datasets • `"cross-encoder-distilroberta"` Uses Cross-Encoder's Natural Language Interface DistilRoBERTa Base zero-shot classification model trained on the Stanford Natural Language Inference (SNLI) corpus and MultiNLI datasets. The DistilRoBERTa is intended to be a smaller, more lightweight version of `"cross-encoder-roberta"`, that sacrifices some accuracy for much faster speed (see https://www.sbert.net/docs/pretrained_cross-encoders.html#nli) • `"facebook-bart"` Uses Facebook's BART Large zero-shot classification model trained on the Multi-Genre Natural Language Inference (MultiNLI) dataset. Defaults to `"cross-encoder-distilroberta"`. Also allows any zero-shot classification models with a pipeline from huggingface to be used by using the specified name (e.g., `"typeform/distilbert-base-uncased-mnli"`; see Examples)
`preprocess`	Boolean. Should basic preprocessing be applied? Includes making lowercase, keeping only alphanumeric characters, removing escape characters, removing repeated characters, and removing white space. Defaults to `FALSE`. Transformers generally are OK without preprocessing and handle many of these functions internally, so setting to `TRUE` will not change performance much
`keep_in_env`	Boolean. Whether the classifier should be kept in your global environment. Defaults to `TRUE`. By keeping the classifier in your environment, you can skip re-loading the classifier every time you run this function. `TRUE` is recommended
`envir`	Numeric. Environment for the classifier to be saved for repeated use. Defaults to the global environment

Value

Returns probabilities for the text classes

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

# BART
Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

# RoBERTa
Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

# Zero-shot classification
Yin, W., Hay, J., & Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. *arXiv preprint arXiv:1909.00161*.

# MultiNLI dataset
Williams, A., Nangia, N., & Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Examples

```
# Load data
data(neo_ipip_extraversion)

# Example text
text <- neo_ipip_extraversion$friendliness[1:5]

## Not run:
# Cross-Encoder DistilRoBERTa
transformer_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 )
)

# Facebook BART Large
transformer_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 ),
 transformer = "facebook-bart"
)
```

```
# Directly from huggingface: typeform/distilbert-base-uncased-mnli
transformer_scores(
 text = text,
 classes = c(
   "friendly", "gregarious", "assertive",
   "active", "excitement", "cheerful"
 ),
 transformer = "typeform/distilbert-base-uncased-mnli"
)

## End(Not run)
```

Index