Package ‘transforEmotion’

January 9, 2024

Title  Sentiment Analysis for Text, Image and Video using Transformer Models

Version  0.1.4

Date  2024-01-08

Maintainer  Aleksandar Tomašević <atomashevic@gmail.com>

Description  Implements sentiment analysis using huggingface <https://huggingface.co> transformer zero-shot classification model pipelines for text and image data. The default text pipeline is Cross-Encoder’s DistilRoBERTa <https://huggingface.co/cross-encoder/nli-distilroberta-base> and default image/video pipeline is Open AI’s CLIP <https://huggingface.co/openai/clip-vit-base-patch32>. All other zero-shot classification model pipelines can be implemented using their model name from <https://huggingface.co/models?pipeline_tag=zero-shot-classification>.

License  GPL (>= 3.0)

Encoding  UTF-8

Imports  reticulate, pbapply, googledrive, LSAfun, dplyr, remotes, Matrix

Suggests  markdown, knitr, rmarkdown, rstudioapi, testthat (>= 3.0.0)

VignetteBuilder  knitr

RoxygenNote  7.2.3

Config/testthat/edition  3

NeedsCompilation  no

Author  Alexander Christensen [aut] (<https://orcid.org/0000-0002-9798-7037>), Hudson Golino [aut] (<https://orcid.org/0000-0002-1601-1447>), Aleksandar Tomašević [aut, cre] (<https://orcid.org/0000-0003-4863-6051>)

Depends  R (>= 3.5.0)

Repository  CRAN

Date/Publication  2024-01-09 12:40:02 UTC
### Description

Implements sentiment and emotion analysis using huggingface transformer zero-shot classification model pipelines on text and image data. The default text pipeline is Cross-Encoder’s DistilRoBERTa and default image/video pipeline is Open AI’s CLIP. 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>, Hudson Golino <hfg9s@virginia.edu>
and Aleksandar Tomasevic <atomashevic@ff.uns.ac.rs>

### References

**calculate_moving_average**

*Calculate the moving average for a time series*

**Description**

This function calculates the moving average for a time series.

**Usage**

```
calculate_moving_average(data, window_size)
```

**Arguments**

- `data` (Matrix or Data frame): The time series data
- `window_size` (Numeric integer): The size of the moving average window.

**Value**

Matrix or Data frame containing the moving average values.

---

**conda_check**

*Check if the "transforEmotion" conda environment exists*

**Description**

This function checks if the "transforEmotion" conda environment exists by running the command "conda env list" and searching for the environment name in the output.

**Usage**

```
conda_check()
```

**Value**

A logical value indicating whether the "transforEmotion" conda environment exists.
dlo_dynamics  

Dynamics function of the DLO model

Description

This function calculates the dynamics of a system using the DLO (Damped Linear Oscillator) model based on Equation 1 (Ollero et al., 2023). The DLO model is a second-order differential equation that describes the behavior of a damped harmonic oscillator. The function takes in the current state of the system, the derivative of the state, the damping coefficient, the time step, and the values of the eta and zeta parameters. It returns the updated derivative of the state.

Usage

dlo_dynamics(x, dxdt, q, dt, eta, zeta)

Arguments

- `x`  Numeric. The current state of the system (value of the latent score).
- `dxdt`  Numeric. The derivative of the state (rate of change of the latent score).
- `q`  Numeric. The damping coefficient.
- `dt`  Numeric. The time step.
- `eta`  Numeric. The eta parameter of the DLO model.
- `zeta`  Numeric. The zeta parameter of the DLO model.

Value

A numeric vector containing the updated derivative of the state.

References


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.
Usages

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 receives a 1, otherwise, 0. Words are only associated with one category.

References


Examples

data("emotions")

Description


Usage

domexicon_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.
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**


**See Also**

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

**Examples**

```r
# 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)
```

**Description**

This function generates and emphasizes the effect of strong emotions expressions during the period where the derivative of the latent variable is high. The observable value of the strongest emotion from the positive or negative group will spike in the next k time steps. The probability of this happening is p at each time step in which the derivative of the latent variable is greater than 0.2. The jump is proportionate to the derivative of the latent variable and the sum of the observable values of the other emotions.
### Usage

```r
emphasize(data, num.observable, num.steps, k = 10, p = 0.5)
```

### Arguments

- **data**: Data frame. The data frame containing the latent and observable variables created by the `simulate_video` function.
- **num.observable**: Numeric integer. The number of observable variables per latent factor.
- **num.steps**: Numeric integer. The number of time steps used in the simulation.
- **k**: Numeric integer. The number of time steps to emphasize the effect of strong emotions on future emotions (default is 10). Alternatively: the length of a strong emotional episode.
- **p**: Numeric. The probability of the strongest emotion being emphasized in the next k time steps (default is 0.5).

### Value

A data frame containing the updated observable variables.

### Description

Function to generate observable data from 2 latent variables (negative and positive affect). The function takes in the latent variable scores, the number of time steps, the number of observable variables per latent factor, and the measurement error variance. It returns a matrix of observable data. The factor loadings are not the same for all observable variables. They have uniform random noise added to them (between -0.15 and 0.15). The loadings are scaled so that the sum of the loadings for each latent factor is 2, to introduce a ceiling effect and to differentiate the dynamics of specific emotions. This is further emphasized by adding small noise to the measurement error variance for each observed variable (between -0.01 and 0.01).

### Usage

```r
generate_observables(X, num.steps, num.obs, error, loadings = 0.8)
```

### Arguments

- **X**: Matrix or Data frame. The (num.steps X 2) matrix of latent variable scores.
- **num.steps**: Numeric integer. Number of time steps.
- **num.obs**: Numeric integer. The number of observable variables per latent factor.
- **error**: Numeric. Measurement error variance.
- **loadings**: Numeric (default = 0.8). The default initial loading of the latent variable on the observable variable.
Value

A (num_steps X num_obs) Matrix or Data frame containing the observable variables.

---

**generate_q**

Generate a matrix of Dynamic Error values for the DLO simulation

**Description**

This function generates a matrix of Dynamic Error values (q) for the DLO simulation.

**Usage**

`generate_q(num_steps, sigma_q)`

**Arguments**

- `num_steps`: Numeric integer. The number of time steps used in the simulation.
- `sigma_q`: Numeric. Standard deviation of the Dynamic Error/

**Value**

A (num_steps X 3) matrix of Dynamic Error values for neutral, negative and positive emotion latent score.

---

**image_scores**

Calculate image scores based on OpenAI CLIP model

**Description**

This function takes an image file and a vector of classes as input and calculates the scores for each class using the OpenAI CLIP model. Primary use of the function is to calculate FER scores - Facial Expression Detection of emotions based on detected facial expression in images. In case there are more than one face in the image, the function will return the scores of the face selected using the face_selection parameter. If there is no face in the image, the function will return NA for all classes. Function uses reticulate to call the Python functions in the image.py file. If you run this package/function for the first time it will take some time for the package to setup a functioning Python virtual environment in the background. This includes installing Python libraries for facial recognition and emotion detection in text, images and video. Please be patient.

**Usage**

`image_scores(image, classes, face_selection = "largest")`
MASS_mvrnorm

Arguments

image The path to the image file or URL of the image.
classes A character vector of classes to classify the image into.
face_selection The method to select the face in the image. Can be "largest" or "left" or "right". Default is "largest" and will select the largest face in the image. "left" and "right" will select the face on the far left or the far right side of the image. Face_selection method is irrelevant if there is only one face in the image.

Value

A data frame containing the scores for each class.

Author(s)

Aleksandar Tomašević <atomashevic@gmail.com>

MASS_mvrnorm

Multivariate Normal (Gaussian) Distribution

Description

This function generates a random sample from the multivariate normal distribution with mean mu and covariance matrix Sigma.

Usage

MASS_mvrnorm(n = 1, mu, Sigma, tol = 1e-06, empirical = FALSE, EISPACK = FALSE)

Arguments

n Numeric integer. The number of observations to generate.
mu Numeric vector. The mean vector of the multivariate normal distribution.
Sigma Numeric matrix. The covariance matrix of the multivariate normal distribution.
tol Numeric. Tolerance for checking the positive definiteness of the covariance matrix.
empirical Logical. Whether to return the empirical covariance matrix.
EISPACK Logical. Whether to use the EISPACK routine instead of the LINPACK routine.

Value

A (n X p) matrix of random observations from the multivariate normal distribution. Updated: 26.10.2023.
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; see costring) 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

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(
plot_sim_emotions

Plot the latent or the observable emotion scores.

Description

Function to plot the latent or the observable emotion scores.

Usage

plot_sim_emotions(df, mode = "latent", title = "")

Arguments

df  Data frame. The data frame containing the latent and observable variables created by the simulate_video function.
mode  Character. The mode of the plot. Can be either 'latent', 'positive' or 'negative'.
title  Character. The title of the plot. Default is an empty title, ' '.

# 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**

**Value**

A plot of the latent or the observable emotion scores.

---

**Description**

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

**Usage**

```r
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**

```r
# Load data
data(neo_ipip_extraversion)

# Example text
text <- neo_ipip_extraversion$friendliness

# Keep only periods
punctuate(text, allowPunctuations = c(".")))```
**setup_miniconda**  
*Install Miniconda and activate the transforEmotion environment*

**Description**  
Installs miniconda and activates the transforEmotion environment

**Usage**  
```
setup_miniconda()
```

**Details**  
Installs miniconda using `install_miniconda` and activates the transforEmotion environment using `use_condaenv`. If the transforEmotion environment does not exist, it will be created using `conda_create`.

**Author(s)**  
Alexander P. Christensen <alexpaulchristensen@gmail.com>  
Aleksandar Tomašević <atomashevic@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>
simulate_video

Simulate latent and observed emotion scores for a single "video"

Description

This function simulates emotions in a video using the DLO model implemented as continuous time state space model. The function takes in several parameters, including the time step, number of steps, number of observables, and various model parameters. It returns a data frame containing the simulated emotions and their derivatives, as well as smoothed versions of the observables. The initial state of the video is always the same. Neutral score is 0.5 and both positive and negative emotion score is 0.25. To simulate more realistic time series, there is an option of including a sudden jump in the emotion scores. This is done by emphasizing the effect of the dominant emotion during the period where the derivative of the latent variable is high. The observable value of the strongest emotion from the positive or negative group will spike in the next k time step (emph.dur). The probability of this happening is p at each time step in which the derivative of the latent variable is greater than 0.2. The jump is proportionate to the derivative of the latent variable and the sum of the observable values of the other emotions.

Usage

simulate_video(
  dt,
  num_steps,
  num_observables,
  eta_n,
  zeta_n,
  eta,
  zeta,
  sigma_q,
  sd_observable,
  loadings,
  window_size,
  emph = FALSE,
  emph.dur = 10,
  emph.prob = 0.5
)

Arguments

dt          Numeric real. The time step for the simulation (in minutes).
num_steps   Numeric real. Total length of the video (in minutes).
num_observables   Numeric integer. The number of observables to generate per factor. Total number of observables generated is 2 x num_observables.
eta_n       Numeric. The eta parameter for the neutral state.
zeta_n      Numeric. The zeta parameter for the neutral state.
**stop_words**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>eta</code></td>
<td>Numeric. The eta parameter for the positive and negative emotions.</td>
</tr>
<tr>
<td><code>zeta</code></td>
<td>Numeric. The zeta parameter for the positive and negative emotions.</td>
</tr>
<tr>
<td><code>sigma_q</code></td>
<td>Numeric. The standard deviation of Dynamic Error of the q(t) function.</td>
</tr>
<tr>
<td><code>sd.observable</code></td>
<td>Numeric. The standard deviation of the measurement error.</td>
</tr>
<tr>
<td><code>loadings</code></td>
<td>Numeric (default = 0.8). The default initial loading of the latent variable on the observable variable.</td>
</tr>
<tr>
<td><code>window_size</code></td>
<td>Numeric integer. The window size for smoothing the observables.</td>
</tr>
<tr>
<td><code>emph</code></td>
<td>Logical. Whether to emphasize the effect of dominant emotion (default is FALSE).</td>
</tr>
<tr>
<td><code>emph.dur</code></td>
<td>Numeric integer. The duration of the emphasis (default is 10).</td>
</tr>
<tr>
<td><code>emph.prob</code></td>
<td>Numeric. The probability of the dominant emotion being emphasized (default is 0.5).</td>
</tr>
</tbody>
</table>

**Value**

A data frame (num_steps X (6 + num_observables)) containing the latent scores for neutral score, positive emotions, negative emotions and their derivatives, as well as smoothed versions of the observables.

**Examples**

```r
simulate_video(dt = 0.01, num_steps = 50, num_observables = 4, 
    eta_n = 0.5, zeta_n = 0.5, 
    eta = 0.5, zeta = 0.5, 
    sigma_q = 0.1, sd.observable = 0.1, 
    loadings = 0.8, window_size = 10)
```

---

**stop_words**

Stop Words from the tm Package

**Description**

174 English stop words in the tm package

**Usage**

```r
data(stop_words)
```

**Format**

A vector (length = 174)

**Examples**

```r
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
transformer_scores

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

Preprocess 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

# RoBERTa

# Zero-shot classification

# MultiNLI dataset

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"
)


video_scores

# 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)

---

**video_scores**  
*Run FER on YouTube video*

**Description**

This function retrieves FER scores a specific number of frames extracted from YouTube video. It uses Python libraries for facial recognition and emotion detection in text, images, and videos.

**Usage**

```r
video_scores(
  video,
  classes,
  nframes = 100,
  face_selection = "largest",
  start = 0,
  end = -1,
  uniform = FALSE,
  ffreq = 15,
  save_video = FALSE,
  save_frames = FALSE,
  save_dir = "temp/",
  video_name = "temp"
)
```

**Arguments**

- **video**: The URL of the YouTube video to analyze.
- **classes**: A character vector specifying the classes to analyze.
- **nframes**: The number of frames to analyze in the video. Default is 100.
- **face_selection**: The method for selecting faces in the video. Options are "largest", "left", or "right". Default is "largest".
- **start**: The start time of the video range to analyze. Default is 0.
end
The end time of the video range to analyze. Default is -1 and this means that video won’t be cut. If end is a positive number greater than start, the video will be cut from start to end.

uniform
Logical indicating whether to uniformly sample frames from the video. Default is FALSE.

ffreq
The frame frequency for sampling frames from the video. Default is 15.

save_video
Logical indicating whether to save the analyzed video. Default is FALSE.

save_frames
Logical indicating whether to save the analyzed frames. Default is FALSE.

save_dir
The directory to save the analyzed frames. Default is "temp/".

video_name
The name of the analyzed video. Default is "temp".

Value
A result object containing the analyzed video scores.

Author(s)
Aleksandar Tomašević <atomashevic@gmail.com>
Index

* datasets
  emotions, 4
  neo_ipip_extraversion, 10
  stop_words, 17
  tinytrolls, 18

calculate_moving_average, 3
conda_check, 3
conda_create, 15
conda_install, 15
costring, 10
dlo_dynamics, 4
emotions, 4, 5, 6
emoxicon_scores, 5
emphasize, 6
generate_observables, 7
generate_q, 8
image_scores, 8
install_miniconda, 15
MASS_mvrnorm, 9
neo_ipip_extraversion, 10
nlp_scores, 10
plot_sim_emotions, 13
punctuate, 14
setup_miniconda, 15
setup_modules, 15
simulate_video, 16
stop_words, 11, 17
tinytrolls, 18
transforEmotion
  (transforEmotion-package), 2
transforEmotion-package, 2
transformer_scores, 15, 18
use_condaenv, 15
video_scores, 21
weights, 5