Package 'zoomGroupStats'

January 20, 2025

Title Analyze Text, Audio, and Video from 'Zoom' Meetings

Version 0.1.0

URL http://zoomgroupstats.org

Description Provides utilities for processing and analyzing the files that are exported from a recorded 'Zoom' Meeting. This includes analyzing data captured through video cameras and microphones, the text-based chat, and meta-data. You can analyze aspects of the conversation among meeting participants and their emotional expressions throughout the meeting.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports data.table, dplyr, lubridate, magick, openxlsx, paws, pbapply, stringr, syuzhet, utils

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Andrew Knight [aut, cre]

Maintainer Andrew Knight <knightap@wustl.edu>

Repository CRAN

Date/Publication 2021-05-13 09:20:02 UTC

Contents

`aggSentiment` *Helper function to aggregate sentiment variables*

Description

Used to aggregate the sentiment variables to the individual and meeting levels

Usage

```
aggSentiment(inputData, meetingId = NULL, speakerId = NULL, sentMethod)
```

Arguments

`inputData`	data.frame that has been output from textSentiment function
`meetingId`	string that indicates the name of the variable containing the meeting ID
`speakerId`	string that indicates the name of the variable containing the speaker identity
`sentMethod`	string that indicates what type of sentiment analysis to aggregate–must be either 'aws' or 'syuzhet'

Value

A data.frame giving the sentiment metrics aggregated to the requested level. If only meetingId is specified, metrics are aggregated to that level. If only speakerId is specified, metrics are aggregated to the individual level across any meetings. If both meetingId and speakerId are specified, metrics are aggregated to the level of the individual within meeting.

Examples

```
agg.out = aggSentiment(inputData=sample_transcript_sentiment_aws,
meetingId="batchMeetingId", speakerId = "userId", sentMethod="aws")

agg.out = aggSentiment(inputData=sample_chat_sentiment_syu,
meetingId="batchMeetingId", speakerId = "userName", sentMethod="syuzhet")
```

---

batchGrabVideoStills *Batch process video files, breaking them into stills*

Description

#' This helper calls grabVideoStills, which function currently relies on the av package and 'ffmpeg' to split a video file into images. This function will save the images to the director specified by the user.

Usage

```
batchGrabVideoStills(
  batchInfo,
  imageDir = NULL,
  overWriteDir = FALSE,
  sampleWindow
)
```

Arguments

batchInfo	the batchInfo data.frame that is output from batchProcessZoomOutput
imageDir	the directory where you want the function to write the extracted image files
overWriteDir	logical indicating whether you want to overwrite imageDir if it exists
sampleWindow	an integer indicating how frequently you want to sample images in number of seconds.

Value

a data.frame that gives information about the batch. Each record corresponds to one video, with:

- batchMeetingId - the meeting identifier
- videoExists - boolean indicating whether the video file was there
- imageDir - path to the directory where video images are saved
- sampleWindow - integer with the sampleWindow requested
- numFramesExtracted - the number of image files that were saved

Examples

```
vidBatchInfo = batchGrabVideoStills(batchInfo=sample_batch_info,
imageDir=tempdir(), overWriteDir=TRUE, sampleWindow=2)
## Not run:
vidBatchInfo = batchGrabVideoStills(batchInfo=zoomOut$batchInfo,
imageDir="~/Documents/myMeetings/videoImages", overWriteDir=TRUE,  sampleWindow=600)

## End(Not run)
```

`batchProcessZoomOutput` *Batch process files that have been downloaded from Zoom*

Description

Provide the location of a structured batchInput file and this function will process a set of meetings at once.

Usage

```
batchProcessZoomOutput(batchInput, exportZoomRosetta = NULL)
```

Arguments

`batchInput`	String giving the location of the xlsx file that contains the information for the zoom meetings. All corresponding Zoom downloads for the meetings in the batch must be saved in the same directory as the batchInput file.
`exportZoomRosetta`	optional string giving the path for exporting the zoomRosetta file to link up unique individual IDs manually. Providing this path will write the zoomRosetta file to that location.

Value

a list that has a data.frame for each of the elements of a Zoom output that are available in the input directory:

- batchInfo - Each row is a meeting included in batchInput. Columns provide information about each meeting.
- meetInfo - Each row is a meeting for which there was a downloaded participants file. Columns provide information about the meeting from the Zoom Cloud recording site.
- partInfo - Each row is a Zoom display name (with display name changes in parentheses). Columns provide information about participants from the Zoom Cloud recording site.
- transcript - Each row is an utterance in the audio transcript. This is the output from processZoomTranscript.
- chat - Each row is a message posted to the chat. This is the output from processZoomChat.
- rosetta - Each row is a unique display name (within meeting) encountered in the batchInput. This is used to reconcile user identities.

Examples

```
batchOut = batchProcessZoomOutput(batchInput=system.file('extdata',
'myMeetingsBatch.xlsx', package = 'zoomGroupStats'),
exportZoomRosetta=file.path(tempdir(),"_rosetta.xlsx"))
```

`batchVideoFaceAnalysis` *Batch analyze faces in videos*

Description

Using this function you can analyze attributes of facial expressions within a batch of video files. This batch approach requires breaking the videos into still frames in advance by using the batchGrabVideoStills() function.

Usage

```
batchVideoFaceAnalysis(
  batchInfo,
  imageDir,
  sampleWindow,
  facesCollectionID = NA
)
```

Arguments

`batchInfo`	the batchInfo data.frame that is output from batchProcessZoomOutput
`imageDir`	the path to the top-level directory of where all the images are stored
`sampleWindow`	an integer indicating how frequently you have sampled images in number of seconds.
`facesCollectionID`	name of an 'AWS' collection with identified faces

Value

data.frame with one record for every face detected in each frame across all meetings. For each face, there is an abundance of information from 'AWS Rekognition'. This output is quite detailed. Note that there will be a varying number of faces per sampled frame in the video. Imagine that you have sampled the meeting and had someone rate each person's face within that sampled moment.

Examples

```
## Not run:
  vidOut = batchVideoFaceAnalysis(batchInfo=zoomOut$batchInfo,
  imageDir="~/Documents/meetingImages",
  sampleWindow = 300)

## End(Not run)
```

---

createZoomRosetta *Create a file to aid in adding a unique identifier to link to the zoom user name*

Description

A major challenge in analyzing virtual meetings is reconciling the display name that zoom users in chat and transcript. This function outputs a data.frame that can be helpful in manually adding a new unique identifier to use in further data anlaysis.

Usage

```
createZoomRosetta(zoomOutput)
```

Arguments

zoomOutput	the output from running processZoomOutput

Value

a data.frame that has unique values for the zoom display name that show up across any files that are available, including participants, transcript, and chat. If the user gives the participants file, it will separate display name changes and include all versions. If there are emails attached to display names, it will include those.

Examples

```
rosetta.out = createZoomRosetta(processZoomOutput(fileRoot=
file.path(system.file('extdata', package = 'zoomGroupStats'),"meeting001")))
## Not run:
rosetta.out = createZoomRosetta(processZoomOutput(fileRoot="~/zoomMeetings/meeting001"))

## End(Not run)
```

grabVideoStills *Helper function to split a video into still frames*

Description

This function currently relies on the av package and 'ffmpeg' to split a video file into images. This function will save the images to the directory specified by the user.

Usage

```
grabVideoStills(
  inputVideo,
  imageDir = NULL,
  overWriteDir = FALSE,
  sampleWindow
)
```

Arguments

inputVideo	full filepath to a video file
imageDir	the directory where you want the function to write the extracted image files
overWriteDir	logical indicating whether you want to overwrite imageDir if it exists
sampleWindow	an integer indicating how frequently you want to sample images in number of seconds.

Value

a data.frame that gives information about the still frames. Each record is a stillframe, with the following info:

- imageSeconds - number of seconds from the start of the video when this image was captured
- imageName - full path to where the image has been saved as a .png

Examples

```
vidOut = grabVideoStills(inputVideo=system.file('extdata', "meeting001_video.mp4",
package = 'zoomGroupStats'), imageDir=tempdir(), overWriteDir=TRUE, sampleWindow=2)
## Not run:
grabVideoStills(inputVideo='myMeeting.mp4',
imageDir="~/Documents/myMeetings/videoImages", overWriteDir=TRUE,  sampleWindow=45)

## End(Not run)
```

importZoomRosetta *Helper function to add unique identifiers to processed Zoom downloads*

Description

Import an edited zoomRosetta file that tells how to link up Zoom display names to some unique individual identifier

Usage

```
importZoomRosetta(zoomOutput, zoomRosetta, meetingId)
```

Arguments

zoomOutput	the output of batchProcessZoomOutput
zoomRosetta	the path to an edited zoomRosetta xlsx
meetingId	the name of the meetingId you want to use

Value

returns zoomOutput with identifiers in zoomRosetta merged to any available data.frames in the zoomOutput file

Examples

```
batchOutIds = importZoomRosetta(zoomOutput=
batchProcessZoomOutput(batchInput=system.file('extdata',
'myMeetingsBatch.xlsx', package = 'zoomGroupStats')),
zoomRosetta=system.file('extdata',
'myMeetingsBatch_rosetta_edited.xlsx', package = 'zoomGroupStats'),
meetingId="batchMeetingId")

## Not run:
batchOutIds = importZoomRosetta(zoomOutput=batchOut, zoomRosetta="myEditedRosetta.xlsx",
meetingId="batchMeetingId")

## End(Not run)
```

makeTimeWindows *Helper function that creates temporal windows in datasets*

Description

This creates a set of temporal windows of specified size so that metrics can be computed within those windows.

Usage

```
makeTimeWindows(inputData, timeVar, windowSize)
```

Arguments

inputData	data.frame that has data over time, usually within a single meeting
timeVar	name of a numeric column that contains the time variable you want to use
windowSize	numeric value giving the length of time window

Value

list with two data.frames:

- windowedData - inputData with the temporal window identifying information included
- allWindows - contains the full set of temporal windows and identifying information. This is valuable because inputData may not have records within all of the possible temporal windows

Examples

```
win.out = makeTimeWindows(sample_transcript_processed,
timeVar="utteranceStartSeconds", windowSize=10)
```

processZoomChat *Process a Zoom chat file*

Description

Parses the data from the chatfile that is downloaded from the Zoom Cloud recording site. Note that this is the file that accompanies a recording. This is not the file that you might download directly within a given Zoom session, nor is it the one that is saved locally on your computer. This is the file that you can access after a session if you record in the cloud.

Usage

```
processZoomChat(
  fname,
  sessionStartDateTime = "1970-01-01 00:00:00",
  languageCode = "en"
)
```

Arguments

fname	String that is the path to the downloaded Zoom .txt chat file
sessionStartDateTime	String that is the start of the session in YYYY-MM-DD HH:MM:SS
languageCode	String denoting the language

Value

data.frame where each record is a message submission in the chat, containing columns:

- messageId - Numeric identifier for each message, only unique within a given meeting
- messageSeconds - When message was posted, in number of seconds from start of session
- messageTime - When message was posted as POSIXct, using the supplied sessionStartDateTime
- userName - Display name of user who posted the message
- message - Text of the message that was posted
- messageLanguage - Language code for the message

Examples

```
ch.out = processZoomChat(
fname=system.file('extdata', "meeting001_chat.txt", package = 'zoomGroupStats'),
sessionStartDateTime = '2020-04-20 13:30:00',
languageCode = 'en')
```

---

processZoomOutput *Wrapper function to process the raw files from Zoom in a single call*

Description

The user provides a fileRoot that is used for a given meeting. Output files should be named as fileRoot_chat.txt; fileRoot_transcript.vtt; and fileRoot_participants.csv. Any relevant files will be processed.

Usage

```
processZoomOutput(
  fileRoot,
  rosetta = TRUE,
  sessionStartDateTime = "1970-01-01 00:00:00",
  recordingStartDateTime = "1970-01-01 00:00:00",
  languageCode = "en"
)
```

Arguments

fileRoot	string giving the path to the files and the root
rosetta	boolean to produce the rosetta file or not
sessionStartDateTime	string giving the start of the session in YYYY-MM-DD HH:MM:SS
recordingStartDateTime	string giving the start of the recording in YYYY-MM-DD HH:MM:SS
languageCode	string giving the language code

Value

a named list containing data.frames for each of the available files:

- meetInfo - A single row with info for the meeting that is in the participants file. Columns provide information about the meeting from the Zoom Cloud recording site.
- partInfo - Each row is a Zoom display name (with display name changes in parentheses). Columns provide information about participants from the Zoom Cloud recording site.
- transcript - Each row is an utterance in the audio transcript. This is the output from processZoomTranscript.
- chat - Each row is a message posted to the chat. This is the output from processZoomChat.
- rosetta - Each row is a unique display name (within meeting) encountered in the batchInput. This is used to reconcile user identities.

Examples

```
zoomOut = processZoomOutput(fileRoot=file.path(
system.file('extdata', package = 'zoomGroupStats'),"meeting001"
), rosetta=TRUE)
## Not run:
zoomOut = processZoomOutput(fileRoot="~/zoomMeetings/myMeeting", rosetta=TRUE)

## End(Not run)
```

processZoomParticipantsInfo *Process participant information from a Zoom meeting export*

Description

This function parses the information from the downloadable meeting information file in Zooms reports section. The function presumes that you have checked the box to include the meeting information in the file. That means that there is a header (2 rows) containing the zoom meeting information. Following that header are four columns: Name of user, user email, total duration, and guest.

Usage

```
processZoomParticipantsInfo(inputPath)
```

Arguments

inputPath	character

Value

list of two data.frames with parsed information from the downloadable Zoom participants file

- meetInfo - provides the meeting level information that Zoom Cloud gives
- partInfo - provides the participant level information that Zoom Cloud gives

Examples

```
partInfo = processZoomParticipantsInfo(
system.file('extdata', "meeting001_participants.csv", package = 'zoomGroupStats')
)
```

processZoomTranscript *Process Zoom transcript file*

Description

Process Zoom transcript file

Usage

```
processZoomTranscript(
  fname,
  recordingStartDateTime = "1970-01-01 00:00:00",
  languageCode = "en"
)
```

Arguments

fname	String that is the path to the exported Zoom .vtt transcript chat file
recordingStartDateTime	String that is the timestamp when the recording was started in YYYY-MM-DD HH:MM:SS
languageCode	String denoting the language

Value

data.frame where each record is an utterance in the transcript, with columns:

- utteranceId - Numeric identifier for each utterance in the transcript
- utteranceStartSeconds - number of seconds from the start of the recording when utterance began
- utteranceStartTime - POSIXct timestamp of the start of the utterance, using recordingStartDateTime as the zero
- utteranceEndSeconds - number of seconds from the start of the recording when utterance ended
- utteranceEndTime - POSIXct timestamp of the end of the utterance, using recordingStartDateTime as the zero
- utteranceTimeWindow - number of seconds that this utterance lasted
- userName - Zoom display name of the person who spoke this utterance
- utteranceMessage - transcribed spoken words of this utterance
- utteranceLanguage - language code for this utterance

Zoom Recording Transcript File Processing

This function parses the data from the transcript file (.vtt) that is downloaded from the Zoom website. NOTE: This is the file that accompanies a recording to the cloud.

Examples

```
tr.out = processZoomTranscript(
fname=system.file('extdata', 'meeting001_transcript.vtt', package = 'zoomGroupStats'),
recordingStartDateTime = '2020-04-20 13:30:00', languageCode = 'en')
```

---

sample_batch_info *Parsed batch info file in a recorded 'Zoom' meeting*

Description

Parsed batch info file in a recorded 'Zoom' meeting

Usage

```
sample_batch_info
```

Format

A data frame with 3 rows of 13 variables:

batchMeetingId a character meeting identification variable

fileRoot the prefix to the files for this particular meeting

participants binary indicating whether there is a participants file downloaded

transcript binary indicating whether there is a transcript file downloaded

chat binary indicating whether there is a chat file downloaded

video binary indicating whether there is a video file downloaded

sessionStartDateTime start of the actual session as a character YYYY-MM-DD HH:MM:SS

recordingStartDateTime start of the actual recording as a character YYYY-MM-DD HH:MM:SS

participants_processed binary indicating whether there is a participants file already processed

transcript_processed binary indicating whether there is a transcript file already processed

chat_processed binary indicating whether there is a chat file already processed

video_processed binary indicating whether there is a video file already processed

dirRoot character giving the directory in which all files will be found

Source

http://zoomgroupstats.org/

---

sample_chat_processed *Parsed chat file in a 'Zoom' meeting*

Description

Parsed chat file in a 'Zoom' meeting

Usage

```
sample_chat_processed
```

Format

A data frame with 30 rows of 9 variables:

batchMeetingId a character meeting identification variable

userName 'Zoom' display name attached to this speaker

messageId an incremented numeric identifier for a marked chat message

messageSeconds when the message was posted as the number of seconds from the start of the recording

messageTime timestamp for message

message text of the message

messageLanguage language code of the message

userEmail character email address

userId numeric id of each speaker

Source

http://zoomgroupstats.org/

`sample_chat_sentiment_aws` *Parsed chat file in a 'Zoom' meeting with sentiment analysis using AWS*

Description

Parsed chat file in a 'Zoom' meeting with sentiment analysis using AWS

Usage

```
sample_chat_sentiment_aws
```

Format

A data frame with 10 rows of 14 variables:

batchMeetingId a character meeting identification variable

messageId an incremented numeric identifier for a marked chat message

userName 'Zoom' display name attached to the messager

messageSeconds when the message was posted as the number of seconds from the start of the recording

messageTime timestamp for message

message text of the message

messageLanguage language code of the message

userEmail character email address

userId numeric id of each speaker

aws_sentClass character giving the sentiment classification of this text

aws_positive probability that this text is mixed emotion

aws_negative probability that this text is negative

aws_neutral probability that this text is neutral

aws_mixed probability that this text is positive

Source

http://zoomgroupstats.org/

`sample_chat_sentiment_syu` *Parsed chat file in a 'Zoom' meeting with sentiment analysis using syuzhet*

Description

Parsed chat file in a 'Zoom' meeting with sentiment analysis using syuzhet

Usage

```
sample_chat_sentiment_syu
```

Format

A data frame with 30 rows of 30 variables:

batchMeetingId a character meeting identification variable

messageId an incremented numeric identifier for a marked chat message

userName 'Zoom' display name attached to the messager

messageSeconds when the message was posted as the number of seconds from the start of the recording

messageTime timestamp for message

message text of the message

messageLanguage language code of the message

userEmail character email address

userId numeric id of each speaker

wordCount number of words in this utterance

syu_anger number of anger words

syu_anticipation number of anticipation words

syu_disgust number of disgust words

syu_fear number of fear words

syu_joy number of joy words

syu_sadness number of sadness words

syu_surprise number of surprise words

syu_trust number of trust words

syu_negative number of negative words

syu_positive number of positive words

Source

http://zoomgroupstats.org/

`sample_transcript_processed` *Parsed spoken language in a 'Zoom' meeting.*

Description

Parsed spoken language in a 'Zoom' meeting.

Usage

```
sample_transcript_processed
```

Format

A data frame with 30 rows of 12 variables:

batchMeetingId a character meeting identification variable

userName 'Zoom' display name attached to this speaker

utteranceId an incremented numeric identifier for a marked speech utterance

utteranceStartSeconds when the utterance started as the number of seconds from the start of the recording

utteranceStartTime timestamp for the start of the utterance

utteranceEndSeconds when the utterance ended as the number of seconds from the start of the recording

utteranceEndTime timestamp for the end of the utterance

utteranceTimeWindow duration of the utterance, in seconds

utteranceMessage the text of the utterance

utteranceLanguage language code of the utterance

userEmail character email address

userId numeric id of each speaker

Source

http://zoomgroupstats.org/

`sample_transcript_sentiment_aws` *Parsed spoken language in a 'Zoom' meeting with AWS-based sentiment analysis.*

Description

Parsed spoken language in a 'Zoom' meeting with AWS-based sentiment analysis.

Usage

```
sample_transcript_sentiment_aws
```

Format

A data frame with 30 rows of 17 variables:

batchMeetingId a character meeting identification variable

utteranceId an incremented numeric identifier for a marked speech utterance

userName 'Zoom' display name attached to this speaker

utteranceStartSeconds when the utterance started as the number of seconds from the start of the recording

utteranceStartTime timestamp for the start of the utterance

utteranceEndSeconds when the utterance ended as the number of seconds from the start of the recording

utteranceEndTime timestamp for the end of the utterance

utteranceTimeWindow duration of the utterance, in seconds

utteranceMessage the text of the utterance

utteranceLanguage language code of the utterance

userEmail character email address

userId numeric id of each speaker

aws_sentClass character giving the sentiment classification of this text

aws_positive probability that this text is mixed emotion

aws_negative probability that this text is negative

aws_neutral probability that this text is neutral

aws_mixed probability that this text is positive

Source

http://zoomgroupstats.org/

`sample_transcript_sentiment_syu` *Parsed spoken language in a 'Zoom' meeting with syuzhet-based sentiment analysis.*

Description

Parsed spoken language in a 'Zoom' meeting with syuzhet-based sentiment analysis.

Usage

```
sample_transcript_sentiment_syu
```

Format

A data frame with 30 rows of 23 variables:

batchMeetingId a character meeting identification variable

utteranceId an incremented numeric identifier for a marked speech utterance

userName 'Zoom' display name attached to this speaker

utteranceStartSeconds when the utterance started as the number of seconds from the start of the recording

utteranceStartTime timestamp for the start of the utterance

utteranceEndSeconds when the utterance ended as the number of seconds from the start of the recording

utteranceEndTime timestamp for the end of the utterance

utteranceTimeWindow duration of the utterance, in seconds

utteranceMessage the text of the utterance

utteranceLanguage language code of the utterance

userEmail character email address

userId numeric id of each speaker

wordCount number of words in this utterance

syu_anger number of anger words

syu_anticipation number of anticipation words

syu_disgust number of disgust words

syu_fear number of fear words

syu_joy number of joy words

syu_sadness number of sadness words

syu_surprise number of surprise words

syu_trust number of trust words

syu_negative number of negative words

syu_positive number of positive words

Source

http://zoomgroupstats.org/

`textConversationAnalysis` *Analyze conversation attributes*

Description

This function takes in the output of one of the other functions (either processZoomChat or processZoomTranscript) and produces a set of conversation measures.

Usage

```
textConversationAnalysis(
  inputData,
  inputType,
  meetingId,
  speakerId,
  sentMethod = "none"
)
```

Arguments

`inputData`	data.frame that is the output of either processZoomChat or processZoomTranscript
`inputType`	string of either 'transcript' or 'chat'
`meetingId`	string giving the name of the variable with the meetingId
`speakerId`	string giving the name of the identifier for the individual who made this contribution
`sentMethod`	string giving the type of sentiment analysis to include, either 'aws' or 'syuzhet'

Value

A list of two data.frames, with names conditional on your choice to analyze a parsed transcript file or a parsed chat file. The first list item contains statistics at the corpus level. The second list item contains statistics at the speaker/messager level of analysis.

Examples

```
convo.out = textConversationAnalysis(inputData=sample_transcript_processed,
inputType='transcript', meetingId='batchMeetingId',
speakerId='userName', sentMethod="none")

convo.out = textConversationAnalysis(inputData=sample_transcript_sentiment_syu,
inputType='transcript', meetingId='batchMeetingId',
```

```
speakerId='userName', sentMethod="syuzhet")

convo.out = textConversationAnalysis(inputData=sample_chat_sentiment_aws,
inputType='chat', meetingId='batchMeetingId',
speakerId='userName', sentMethod="aws")

## Not run:
convo.out = textConversationAnalysis(inputData=sample_transcript_sentiment_aws,
inputType='transcript', meetingId='batchMeetingId',
speakerId='userName', sentMethod="aws")

convo.out = textConversationAnalysis(inputData=sample_transcript_sentiment_syu,
inputType='transcript', meetingId='batchMeetingId',
speakerId='userName', sentMethod="syuzhet")

convo.out = textConversationAnalysis(inputData=sample_chat_processed,
inputType='chat', meetingId='batchMeetingId',
speakerId='userName', sentMethod="none")

convo.out = textConversationAnalysis(inputData=sample_chat_sentiment_aws,
inputType='chat', meetingId='batchMeetingId',
speakerId='userName', sentMethod="aws")

convo.out = textConversationAnalysis(inputData=sample_chat_sentiment_syu,
inputType='chat',meetingId='batchMeetingId',
speakerId='userName', sentMethod="syuzhet")

## End(Not run)
```

`textSentiment` *Conduct a sentiment analysis on text data*

Description

This function takes in the output of the chat and transcript functions. It then conducts a sentiment analysis on an identified chunk of text and returns the values. To use the aws option, you must have an aws account that with privileges for the comprehend service However you authenticate for AWS, you should do so before running calling the function with this option in sentMethods

Usage

```
textSentiment(
  inputData,
  idVars,
  textVar,
  sentMethods,
  appendOut = FALSE,
  languageCodeVar
)
```

Arguments

inputData	data.frame that has been output by either the processZoomTranscript or processZoomChat functions
idVars	vector with the name of variables that give the unique identifiers for this piece of text. Usually this will be a the meeting id variable and the text id variable (e.g., utteranceId, messageId)
textVar	name of variable that contains the text
sentMethods	a vector specifying the types of sentiment analysis-currently either "aws" or "syuzhet"
appendOut	boolean indicating whether you want the sentiment results merged to the inputData in your output
languageCodeVar	name of variable that contains the language code

Value

returns a list containing as data.frames the output of the sentiment analyses that were requested in sentMethods. For each output data.frame, the first columns are the idVars specified to enable combining back with the original inputData

Examples

```
sent.out = textSentiment(inputData=sample_chat_processed,
idVars=c('batchMeetingId', 'messageId'),
textVar='message', sentMethods='syuzhet',appendOut=TRUE,
languageCodeVar='messageLanguage')

## Not run:
sent.out = textSentiment(inputData=sample_transcript_processed,
idVars=c('batchMeetingId','utteranceId'),
textVar='utteranceMessage', sentMethods=c('aws','syuzhet'),
appendOut=TRUE, languageCodeVar='utteranceLanguage')

## End(Not run)
```

---

turnTaking *Simple conversational turn-taking analysis*

Description

Generate a very basic analysis of the conversational turntaking in either a Zoom transcript or a Zoom chat file.

Usage

```
turnTaking(inputData, inputType, meetingId, speakerId)
```

Arguments

inputData	data.frame output from either processZoomChat or processZoomTranscript
inputType	string of either 'chat' or 'transcript'
meetingId	string giving the name of the meeting identifier
speakerId	string giving the name of the variable with the identity of the speaker

Value

list of four data.frames giving different levels of analysis for turn taking:

- rawTurn - This data.frame gives a dataset with a lagged column so that you could calculate custom metrics
- aggTurnsDyad - This gives a dyad-level dataset so that you know whose speech patterns came before whose
- aggTurnsSpeaker - This gives a speaker-level dataset with metrics that you could use to assess each given person's influence on the conversation
- aggTurnsSpeaker_noself - This is a replication of the aggTurnsSpeaker dataset, but it excludes turns where a speaker self-follows (i.e., Speaker A => Speaker A)

Examples

```
turn.out = turnTaking(inputData=sample_transcript_processed,
inputType='transcript', meetingId='batchMeetingId',
speakerId='userName')

turn.out = turnTaking(inputData=sample_chat_processed,
inputType='chat', meetingId='batchMeetingId',
speakerId='userName')
```

---

videoFaceAnalysis *Analyze the facial features within an exported Zoom video file*

Description

Using this function you can analyze attributes of facial expressions within a video file. There are two ways to supply the video information. First, you can provide the actual video file. The function will then break it down into still frames using the grabVideoStills() function. Second, you can use the videoImageDirectory argument to give the location of a directory where images have been pre-saved.

Usage

```
videoFaceAnalysis(
  inputVideo,
  recordingStartDateTime,
  sampleWindow,
  facesCollectionID = NA,
  videoImageDirectory = NULL,
  grabVideoStills = FALSE,
  overWriteDir = FALSE
)
```

Arguments

inputVideo	string path to the video file (ideal is gallery)
recordingStartDateTime	YYYY-MM-DD HH:MM:SS of the start of the recording
sampleWindow	Frame rate for the analysis
facesCollectionID	name of an 'AWS' collection with identified faces
videoImageDirectory	path to a directory that either contains image files or where you want to save image files
grabVideoStills	logical indicating whether you want the function to split the video file or not
overWriteDir	logical indicating whether to overwrite videoImageDirectory if it exists

Value

data.frame with one record for every face detected in each frame. For each face, there is an abundance of information from 'AWS Rekognition'. This output is quite detailed. Note that there will be a varying number of faces per sampled frame in the video. Imagine that you have sampled the meeting and had someone rate each person's face within that sampled moment.

Examples

```
## Not run:
vid.out = videoFaceAnalysis(inputVideo="meeting001_video.mp4",
recordingStartDateTime="2020-04-20 13:30:00",
sampleWindow=1, facesCollectionID="group-r",
videoImageDirectory="~/Documents/meetingImages",
grabVideoStills=FALSE, overWriteDir=FALSE)

## End(Not run)
```

windowedTextConversationAnalysis *Run a windowed analysis on either a Zoom transcript or chat This function conducts a temporal window analysis on the conversation in either a Zoom transcript or chat. It replicates the textConversationAnalysis function across a set of windows at a window size specified by the user.*

Description

Run a windowed analysis on either a Zoom transcript or chat This function conducts a temporal window analysis on the conversation in either a Zoom transcript or chat. It replicates the textConversationAnalysis function across a set of windows at a window size specified by the user.

Usage

```
windowedTextConversationAnalysis(
  inputData,
  inputType,
  meetingId,
  speakerId,
  sentMethod = "none",
  timeVar = "automatic",
  windowSize
)
```

Arguments

inputData	data.frame output of either processZoomTranscript or processZoomChat
inputType	string of either 'chat' or 'transcript'
meetingId	string giving the column with the meeting identifier
speakerId	string giving the name of the identifier for the individual who made this contribution
sentMethod	string giving the type of sentiment analysis to include, either 'aws' or 'syuzhet'
timeVar	name of variable giving the time marker to be used. For transcript, either use 'utteranceStartSeconds' or 'utteranceEndSeconds'; for chat use 'messageTime'
windowSize	integer value of the duration of the window in number of seconds

Value

list with two data.frames. In the first (windowlevel), each row is a temporal window. In the second (speakerlevel), each row is a user's metrics within a given temporal window.

Examples

```
win.text.out = windowedTextConversationAnalysis(inputData=sample_transcript_sentiment_aws,
inputType="transcript", meetingId="batchMeetingId", speakerId="userName", sentMethod="aws",
timeVar="utteranceStartSeconds", windowSize=600)
```

Index