2R Assignment 2 Documentation

Release 2025

Constantina Maltezou

Sep 01, 2025

1 Introduction

This is the documentation for my second class assignment **2R Analysing Multimodal Language Data for Quantitative Social Science**.

1. **custom_packages.misc_modules.config**
2. **custom_packages.misc_modules.styling_and_animations**
3. **custom_packages.misc_modules.visualisations**

The code provided here is designed to:

- **Preprocess** the CSV data relating to extracted acoustic features and speaker demographics.
- **Classify** speech based on its intent (trustworthy vs. neutral).
- **Validate** the dataset's effectiveness in categorising speech according to speaker intent and demographics.
- **Visualise** the relationship between speaker demographics, acoustic features and trustworthy intent classifications.

For detailed instructions on how to set up the environment, run the analysis scripts, and interpret the results, refer to the **Getting Started** section.

2 Section 2

2.1 Module: custom_packages.misc_modules.config

get_openai_client
get_gemini_client
get_youtube_client

custom_packages.misc_modules.config.get_openai_client

get_openai_client()

custom_packages.misc_modules.config.get_gemini_client

get_gemini_client()

custom_packages.misc_modules.config.get_youtube_client

get_youtube_client()

Module: custom_packages.misc_modules.styling_and_animations

print_error_message	Displaying a red-coloured error message in the console.
print_success_message	Displaying a green-coloured success message in the console.
print_warning_message	Displaying an orange-coloured warning message in the console.
print_highlighted_message	Displaying a yellow-highlighted message in the console.
print_animated_ellipsis_message	Displaying a message in the console, followed by a simple, animated ellipsis to indicate progress.

custom_packages.misc_modules.styling_and_animations.print_error_message

print_error_message(*message*, *e*)

Displaying a red-coloured error message in the console.

Parameters

- **message** (*str*)
- **e** (*Exception*)

Return type

None

custom_packages.misc_modules.styling_and_animations.print_success_message

print_success_message(*message*)

Displaying a green-coloured success message in the console.

Parameters

message (*str*)

Return type

None

custom_packages.misc_modules.styling_and_animations.print_warning_message

print_warning_message(*message*)

Displaying an orange-coloured warning message in the console.

Parameters

message (*str*)

Return type

None

custom_packages.misc_modules.styling_and_animations.print_highlighted_message

print_highlighted_message(*message*)

Displaying a yellow-highlighted message in the console.

Parameters

message (*str*)

Return type

None

custom_packages.misc_modules.styling_and_animations.print_animated_ellipsis_message

print_animated_ellipsis_message(*message='Currently processing'*, *duration=4.0*, *sleep_interval=0.2*)

Displaying a message in the console, followed by a simple, animated ellipsis to indicate progress.

Parameters

- **message** (*str, optional*) – The message to be displayed in the console. The default is "Currently processing".
- **duration** (*float, optional*) – Indicates the total duration (seconds) of the animated ellipsis. The default is 4.0.
- **sleep_interval** (*float, optional*) – Indicating how many seconds needed before each ellipsis update. The default is 0.2.

Return type

None

Module: custom_packages.misc_modules.visualisations

plot_word_frequencies	Shows a bar plot with the most frequent words across all transcripts.
word_cloud_visualisation	Visualises a word cloud of the top keywords across all transcripts.
plot_sentiment_distribution	Shows a bar plot of the count distribution of sentiment labels across all transcripts.
plot_trust_distribution	Shows a bar plot of the count distribution of trust labels across all transcripts.
plot_topic_clusters	Shows a scatter plot that groups transcripts with similar topics.
plot_sentiment_by_topic	Shows a bar plot that shows the distribution of trust scores from a model for each topic category.

custom_packages.misc_modules.visualisations.plot_word_frequencies

plot_word_frequencies(*df*, *words_num*)

Shows a bar plot with the most frequent words across all transcripts.

Parameters

- **df** (*pd.DataFrame*) – A document-ffeature matrix (DFM) with keyword frequencies.
- **words_num** (*int*) – How many highest-frequency keywords to show in the plot.

Return type

None

custom_packages.misc_modules.visualisations.word_cloud_visualisation

word_cloud_visualisation(*df*, *words_num*)

Visualises a word cloud of the top keywords across all transcripts.

Parameters

- **df** (*pd.DataFrame*) – A document-ffeature matrix (DFM) with keyword frequencies.
- **words_num** (*int*) – How many highest-frequency keywords to show in the word cloud.

Return type

None

custom_packages.misc_modules.visualisations.plot_sentiment_distribution

plot_sentiment_distribution(*df*, *sentiment_column*)

Shows a bar plot of the count distribution of sentiment labels across all transcripts.

Parameters

- **df** (*pandas.DataFrame*)
- **sentiment_column** (*str*)

Return type

None

custom_packages.misc_modules.visualisations.plot_trust_distribution

plot_trust_distribution(*df*, *sentiment_column*)

Shows a bar plot of the count distribution of trust labels across all transcripts.

Parameters

- **df** (*pandas.DataFrame*)
- **sentiment_column** (*str*)

Return type

None

custom_packages.misc_modules.visualisations.plot_topic_clusters

plot_topic_clusters(*transcript_topic_distribution*, *dominant_topic_labels*)

Shows a scatter plot that groups transcripts with similar topics.

Parameters

- **transcript_topic_distribution** (*numpy.ndarray*)
- **dominant_topic_labels** (*numpy.ndarray*)

Return type

None

custom_packages.misc_modules.visualisations.plot_sentiment_by_topic

plot_sentiment_by_topic(*df*, *trust_column_name*, *topic_column_name*)

Shows a bar plot that shows the distribution of trust scores from a model for each topic category.

Parameters

- **df** (*pandas.DataFrame*)
- **trust_column_name** (*str*)
- **topic_column_name** (*str*)

Return type

None

2.2 Module: custom_packages.preprocessing_modules.data_collector

store_all_video_data	Uses the given dictionary of search query strings to retrieve relevant Youtube videos, and store them locally as json files.
get_youtube_video_data	
get_json_data	Get all json files from given directory and return a subset of their data in a dataframe format.
get_transcripts	Get all transcript .txt files from given directory and combine their text in a single string output.
get_cleaned_csv_record	Retrieves a CSV file as a dataframe from the given path.

custom_packages.preprocessing_modules.data_collector.store_all_video_data

store_all_video_data(*search_query_groups*, *max_results=10*)

Uses the given dictionary of search query strings to retrieve relevant Youtube videos, and store them locally as json files.

Parameters

- **search_query_groups** (*dict*) – A dictionary of strings, for all search queries per search group (e.g., trust/distrust groups).
- **max_results** (*int, optional*) – The maximum number of total video results to retrieve and store. The default is 10.

Raises

ValueError – Raised if no Youtube data retrieved / empty dataframe.

Returns

Contains all the extracted video data.

Return type

pd.DataFrame

custom_packages.preprocessing_modules.data_collector.get_youtube_video_data

get_youtube_video_data(*query*, *query_group_name*, *max_results=10*)

1. Uses Youtube API v3 to retrieve the list of relevant videos based on the function parameters
2. Extract a list of video data to return.
3. Store all of the retrieved search data locally as a json file on the computer.

Parameters

- **query** (*str*) – The search term to lookup on Youtube.
- **query_group_name** (*str*) – A unique identifier for this query - used in the exported json filename.
- **max_results** (*int (optional; default value of 10)*) – Specifies specifies the maximum number of items(video) that should be returned.

Returns

Contains all the extracted video data.

Return type

pd.Dataframe

custom_packages.preprocessing_modules.data_collector.get_json_data

get_json_data(*files_dir*)

Get all json files from given directory and return a subset of their data in a dataframe format.

Parameters

files_dir (*Path.PosixPath*) – The directory where all the json files are stored.

Returns

Contains json data such as video_id, url, publish_time, etc.

Return type

pd.DataFrame

custom_packages.preprocessing_modules.data_collector.get_transcripts

get_transcripts(*files_dir*, *platform_name*)

Get all transcript .txt files from given directory and combine their text in a single string output.

Parameters

- **files_dir** (*Path.PosixPath*) – The directory where all the transcription .txt files are stored.
- **platform_name** (*str*)

Returns

Contains the combined text from all the .txt files.

Return type

str

custom_packages.preprocessing_modules.data_collector.get_cleaned_csv_record

get_cleaned_csv_record(*csv_filepath*)

Retrieves a CSV file as a dataframe from the given path.

Parameters

csv_filepath (*Path*)

Return type

pandas.DataFrame

2.3 Module: custom_packages.preprocessing_modules.text_cleaning

clean_transcripts	Cleans transcripts per platform and returns a dataframe where each row contains a unique transcript identifier, the actual filename retrieved from Path, the platform name, the raw text, and the cleaned text.
clean_text	Preprocesses the string argument, by stripping unecessary white spaces, converts to lowercase, decouples contractions and replaces dashes with an empty space.
store_cleaned_data_as_csv	Extracts the raw and cleaned data from the dataframe argument and stores them in the corresponding directory offered as an argument.
store_as_csv	Stores the dataframe argument as a CSV file in the given directory.

custom_packages.preprocessing_modules.text_cleaning.clean_transcripts

clean_transcripts(*filepaths_per_platform*)

Cleans transcripts per platform and returns a dataframe where each row contains a unique transcript identifier, the actual filename retrieved from Path, the platform name, the raw text, and the cleaned text.

Parameters

filepaths_per_platform (*dict[str, list[Path]]*) – Contains the platform name (e.g., youtube) as key, and the full paths to the relevant transcripts as the value.

Return type

pd.DataFrame

custom_packages.preprocessing_modules.text_cleaning.clean_text

clean_text(*transcript_text*)

Preprocesses the string argument, by stripping unecessary white spaces, converts to lowercase, decouples contractions and replaces dashes with an empty space.

Parameters

transcript_text (*str*) – The text to be preprocessed.

Returns

The cleaned text after preprocessing.

Return type

str

custom_packages.preprocessing_modules.text_cleaning.store_cleaned_data_as_csv

store_cleaned_data_as_csv(*transcript_records*, *cleaned_data_dir*, *raw_data_dir*)

Extracts the raw and cleaned data from the dataframe argument and stores them in the corresponding directory offered as an argument.

Parameters

- **transcript_records** (*pandas.DataFrame*)
- **cleaned_data_dir** (*Path*)
- **raw_data_dir** (*Path*)

Return type

None

custom_packages.preprocessing_modules.text_cleaning.store_as_csv

store_as_csv(*df*, *full_filepath*)

Stores the dataframe argument as a CSV file in the given directory.

Parameters

- **df** (*pandas.DataFrame*)
- **full_filepath** (*Path*)

Return type

None

2.4 Module: custom_packages.preprocessing_modules.transcriber

transcribe_all_videos	Initiates the transcription process by retrieving the id and url of each video and passing it on to the audio transcription service.
process_audio_stream	
transcribe_youtube_audio_from_videos	
check_file_size_exceeded_limit	Check if the file size in the given path exceeds the expected maximum size limit.
subset_audio_with_overlap	Segment audio files (with pydub) larger than 25MB into smaller segments, to support OpenAI Whisper's transcription.
transcribe_audio	Retrieves the audio file from the given filepath, transcribes it using the whisper-1 model, and stores it based on the video id as a .txt file in the given filepath.
transcribe_segmented_audio	Retrieves the audio segment files from the given filepath based on the video id, transcribes them using the whisper-1 model, and stores them as a concatenated string as a .txt file in the given filepath with the video id argument as part of the transcript name.
store_transcript	Stores the transcribed text as a .txt file in the given filepath, using video id as part of the .txt filename.

custom_packages.preprocessing_modules.transcriber.transcribe_all_videos

transcribe_all_videos(*video_data*)

Initiates the transcription process by retrieving the id and url of each video and passing it on to the audio transcription service.

Parameters

video_data (*pd.Dataframe*) – Contains json youtube data such as the video url, id, published_date in a dataframe format for easier data manipulation.

Returns

The combined text across all transcription files.

Return type

str

custom_packages.preprocessing_modules.transcriber.process_audio_stream

process_audio_stream(*video_url*)

1. Downloading the best audio stream, starting from m4a if available, as its small size is convinient for my local resources. 2. Converting that audio stream to a mono 16kHz mp4 audio file. 3. Storing that audio file as a temporary resource.

Parameters

video_url (*str*) – Used to identify and retrieve the online video from Youtube.

Returns

The path directory of the temporarily stored mp4 audio file. If failed, returns empty string.

Return type

str

custom_packages.preprocessing_modules.transcriber.transcribe_youtube_audio_from_videos

transcribe_youtube_audio_from_videos(*video_id*, *video_url*, *trim_duration_secs=2400*)

1. Retrieving the Youtube video based on the URL in an m4a format.
2. Using OpenAI's whisper-1 model to transcribe all temporary-stored wav audio files.

Parameters
- **video_id** (*str*) – Used to mark the transcription file for identification purposes.
- **video_url** (*str*) – Used to identify and retrieve the online video from Youtube.

Returns
The full text of the transcription.

Return type
String

custom_packages.preprocessing_modules.transcriber.check_file_size_exceeded_limit

check_file_size_exceeded_limit(*path*, *max_size_limit*)

Check if the file size in the given path exceeds the expected maximum size limit.

Parameters
- **path** (*str*) – The path where the file is located.
- **max_size** (*float*) – The maximum size limit permitted for this file.
- **max_size_limit** (*float*)

Returns
True if the size of the file exceeds the indicated maximum value; False if within the maximum size limit.

Return type
bool

custom_packages.preprocessing_modules.transcriber.subset_audio_with_overlap

subset_audio_with_overlap(*audio_id*, *audio_file_path*, *output_dir*, *chunk_length_ms=2400000*, *overlap_ms=1000*)

Segment audio files (with pydub) larger than 25MB into smaller segments, to support OpenAI Whisper's transcription.

Parameters
- **video_id** (*str*) – Used to mark the smaller audio file segments for identification purposes.
- **audio_file_path** (*str*) – To locate and retrieve the large audio file to be segmented.
- **output_dir** (*str*) – Location path to save all smaller audio file segments.
- **chunk_length_ms** (*int (optional)*) – How long, in milliseconds, each segment duration should be. Default is 2.4m milliseconds (i.e., 40 minutes). 1 minute = 60k milliseconds.
- **overlap_ms** (*int*) – How many milliseconds should the overlap be between segments.
- **audio_id** (*str*)

Returns
True if successfully segmented and stored in the output_dir. Default is False.

Return type
Boolean

custom_packages.preprocessing_modules.transcriber.transcribe_audio

transcribe_audio(*video_id*, *audio_filepath*, *transcript_filepath*)

Retrieves the audio file from the given filepath, transcribes it using the whisper-1 model, and stores it based on the video id as a .txt file in the given filepath.

Parameters

- **video_id** (*str*)
- **audio_filepath** (*str*)
- **transcript_filepath** (*str*)

Return type

str

custom_packages.preprocessing_modules.transcriber.transcribe_segmented_audio

transcribe_segmented_audio(*video_id*, *audio_dir*, *transcript_dir*)

Retrieves the audio segment files from the given filepath based on the video id, transcribes them using the whisper-1 model, and stores them as a concatenated string as a .txt file in the given filepath with the video id argument as part of the transcript name.

Parameters

- **video_id** (*str*)
- **audio_dir** (*Path*)
- **transcript_dir** (*str*)

Return type

str

custom_packages.preprocessing_modules.transcriber.store_transcript

store_transcript(*video_id*, *transcript_text*, *transcript_filepath*)

Stores the transcribed text as a .txt file in the given filepath, using video id as part of the .txt filename.

Parameters

- **video_id** (*str*) – The id of the video's transcript to be used as part of the .txt filename.
- **transcript_text** (*str*) – The full transcribed text.
- **transcript_filepath** (*str*) – The full path to store the .txt transcript.

Returns

True if successfully stored locally, otherwise False.

Return type

bool

2.5 Module: custom_packages.modelling_modules.nlp_modelling

get_document_feature_matrix	Creates a basic bag-of-words document-feature matrix (DFM) from the given cleaned text in the dataframe.
sentiment_analysis_using_nltk	
sentiment_analysis_using_distilbert	
sentiment_analysis_using_gpt	
sentiment_analysis_using_google	
sentiment_model_comparison	The dataframe argument should contain the trust sentiment values from all models.
print_models_agreement_percentage	Calculates and displays in the console the agreement rate between two models.
map_sentiment_to_trust	Converts classic sentiment labels to trust/distrust/neutral.
get_dominant_topic	Does topic modelling using FASTopic on the cleaned text from each transcript and retrieves teh top keywords identified per topic (5 most popular topics in this instance).

custom_packages.modelling_modules.nlp_modelling.get_document_feature_matrix

get_document_feature_matrix(*df*)

Creates a basic bag-of-words document-feature matrix (DFM) from the given cleaned text in the dataframe.

Parameters
df (*pd.DataFrame*) – Needs to contain a column named "cleaned_text".

Returns
dfm

Return type
pd.DataFrame

custom_packages.modelling_modules.nlp_modelling.sentiment_analysis_using_nltk

sentiment_analysis_using_nltk(*df*)

1. **Retrieves the partly cleaned/preprocessed text (decoupling contranctions, converting to lowercase and removing**
   dashes and unnecessary white spaces already done earlier) from the dataframe argument.
2. Finishes preprocessing by removing punctuation, stopwords, and applies tokenisation through NLTK.
3. **Does sentiment analysis using NLTK Vader SentimentIntensityAnalyzer on the tokenised text, and calculates the**
   polarity scores (positive, negative, neutral).
4. Visualises the sentiment and translated trust labels in a plot each.
5. Returns a dataframe with the scores and labels.

Parameters
df (*pandas.DataFrame*)

Return type
pandas.DataFrame

custom_packages.modelling_modules.nlp_modelling.sentiment_analysis_using_distilbert

sentiment_analysis_using_distilbert(*df*)

1. **Retrieves the partly cleaned/preprocessed text (decoupling contranctions, converting to lowercase and removing**
   dashes and unnecessary white spaces already done earlier) from the dataframe argument. No need for any further preprocessing nor tokenisation as DistilBert handles that internally.
2. **Does sentiment analysis using the DistilBert sentiment pipeline and calculates the polarity scores** (positive, negative).
3. Visualises the sentiment and translated trust labels in a plot each.
4. Returns a dataframe with the scores and labels.

Parameters
df (*pandas.DataFrame*)
Return type
None

custom_packages.modelling_modules.nlp_modelling.sentiment_analysis_using_gpt

sentiment_analysis_using_gpt(*df*)

1. **Retrieves the partly cleaned/preprocessed text (decoupling contranctions, converting to lowercase and removing**
   dashes and unnecessary white spaces already done earlier) from the dataframe argument. No need for any further preprocessing nor tokenisation as gpt-4o-mini handles that internally.
2. **Does sentiment analysis using the gpt-4o-mini model (requires OpenAI API) and responds with json-formatted**
   results on polarity scores (positive, negative, neutral).
3. Visualises the sentiment and translated trust labels in a plot each.
4. Returns a dataframe with the scores, labels, and summary highlights.

Parameters
df (*pandas.DataFrame*)
Return type
pandas.DataFrame

custom_packages.modelling_modules.nlp_modelling.sentiment_analysis_using_google

sentiment_analysis_using_google(*df*)

1. **Retrieves the partly cleaned/preprocessed text (decoupling contranctions, converting to lowercase and removing**
   dashes and unnecessary white spaces already done earlier) from the dataframe argument. No need for any further preprocessing nor tokenisation as gemini-2.0-flash-lite handles that internally.
2. **Does sentiment analysis using the gemini-2.0-flash-lite model (requires Google API) and responds with**
   json-formatted results on polarity scores (positive, negative, neutral).
3. Visualises the sentiment and translated trust labels in a plot each.
4. Returns a dataframe with the scores, labels, and summary highlights.

Parameters
df (*pandas.DataFrame*)
Return type
pandas.DataFrame

custom_packages.modelling_modules.nlp_modelling.sentiment_model_comparison

sentiment_model_comparison(*df*)

The dataframe argument should contain the trust sentiment values from all models. Compares and displays in the console the agreement rate between all models.

Parameters

df (*pandas.DataFrame*)

Return type

None

custom_packages.modelling_modules.nlp_modelling.print_models_agreement_percentage

print_models_agreement_percentage(*first_model*, *second_model*)

Calculates and displays in the console the agreement rate between two models.

Parameters

- **first_model** (*pandas.Series*)
- **second_model** (*pandas.Series*)

Return type

None

custom_packages.modelling_modules.nlp_modelling.map_sentiment_to_trust

map_sentiment_to_trust(*sentiment_label*)

Converts classic sentiment labels to trust/distrust/neutral.

Parameters

sentiment_label (*str*)

Return type

str

custom_packages.modelling_modules.nlp_modelling.get_dominant_topic

get_dominant_topic(*df*)

Does topic modelling using FASTopic on the cleaned text from each transcript and retrieves teh top keywords identified per topic (5 most popular topics in this instance). Displays a plot per sentiment model for a trust-sentiment score by topic assessment. It also displays another plot that clusters transcripts with similar topics.

Parameters

df (*pandas.DataFrame*)

Return type

pandas.DataFrame