Replication Guide – Tweets Extraction and GPT classification

September 15, 2025

1 Purpose

This guide explains, step-by-step, how to extract data using Twitter historical APIv2 that will be used as the raw data for the main analysis of the paper. It also details the procedure for fetching and classifying, according to the drama triangle, tweets using OpenAI's language models.
This document is organized as follows. Section 2 describes the process and the requirements to extract tweets and prepare the dataset that serves as the input for the GPT annotation part. Section 3 describes the annotation process using OpenAI's API.

2 Data Collection and Preparation

This section explains how to extract tweets using Twitter historical APIv2, and prepare them for the annotation process using OpenAI API. The extraction for our analysis had been done between 7 and 8 February 2022, and between 4 and 7 December 2022. Unfortunately, the historical API used is not open access anymore. Replication of this process can't be done without a subscription. If subscription is available, further adjustments to the code might be necessary to match differences in the source. Here a table summarizing the three main scripts detailed below:

2.1 01_tweet_extraction.py

This script is the core of the data extraction process. It allows the user to query Twitter historical APIv2 with specific keywords, in some selected days. We download tweets covering the 2010-2021 period during random days and during every Saturday in this period. Information about keywords selection can be found in Paper Appendix A1. The outputs of this script are JSON files with raw tweets.

Required Software: prior to replicate this script run the following commands in Anaconda Prompt:

```
conda env create -f acn_data_manage.yml
conda activate acn_data_manage
```

The requirements file `acn_data_manage.yml` contains the Python version to use (3.9.10), and the packages to install.

Lastly, to access the Twitter historical API it is necessary a valid Bearer Token. This has to be set into the script, setting `os.environ['BEARER TOKEN'] = 'your toke here'`.

Folder Structure: to run properly this script, the following folder structure is required:

```
climate_change_narratives/     # root_dir
 input/
   python/                     # currently not used, but the function defines it
                                 (can stay empty, or put helper scripts here)
   mTurk/
      tweets/                  # (not used in this script, but defined)
```

```
output/
   data/
      original_tweets/
         random_days/          # JSON output for random days
         every_saturday/       # JSON output for Saturday queries
```

Map: Input → Output: no external input files are required. The only inputs are hardcoded in the script, and these are the list of random dates, the list of saturdays, the keyword list that form the Twitter API Query, and the bearer token. Outputs are stored in `output/data/original_tweets/random_days/`and `output/data/original_tweets/saturday/`.

2.2 02_data_prep.py

This script processes raw Twitter data and prepares it for downstream analysis and manual annotation. It is the second step in the workflow, following the extraction of tweets via the Twitter Historical Search API. It takes as inputs the raw tweets (output of `01_tweet_extraction.py`) and cleans them. The script extracts and flattens metadata, cleans and standardizes tweet text, removes duplicates, and generates derived variables such as word count and hashtag/mention counts. It outputs cleaned CSV files per day and two final, fully processed datasets, one for random days and one for Saturdays.

Required Software: prior to replicate this script run the same commands as in Section 2.1.

Folder Structure: to run properly this script, the following folder structure is required (if you already have the structure specified in Section 2.1, add only the commented folders):

```
climate_change_narratives/
 input/
   python/
     lookups/                  # to be added
        Emoji_Dict.p           # required pickle used by convert_emojis_to_word()
   visibrain_data/             # to be added
 output/
   data/
      original_tweets/
        random_days/           # JSONs from extraction (input to this script)
        every_saturday/        # JSONs from extraction (input to this script)
      processed_tweets/
         random_days/          # to be added (output directory)
         every_saturday/       # to be added (output directory)
         aggregated_data/      # to be added (output directory)
```

Map: Input → Output: the input of this code is the output of Section 2.1, namely the raw tweets in JSON format. Moreover, an important input is the custom dictionary `Emoji_dict.p`. It has to be stored in `input/python/lookups/`.
Outputs are stored in the output directories pointed out in the folder structure above.

2.3 03_pgeocode_usa.py

This script assigns geographic locations to tweets based on the self-reported or geotagged user locations. It is used to identify whether a tweet originates from within the United States and at what resolution (e.g., state, city). The process involves several steps; first, it extracts all potential locations from user profile data and geotag in two tweet datasets, then it geocodes these locations using the Nominatim API (OpenStreetMap) in 500-location batches. Next, it converts geocoded coordinates into US states using a shapefile and spatial join. It labels each tweet with its corresponding US location, and finally it creates

the output as a final merged dataset of US-based tweets, ready for the GPT annotation process.
It takes as input the processed files from Section 2.2, and the GADM shapefile for USA (level 1).

Required Software: prior to replicate this script run the following commands in Anaconda Prompt. The requirements file `acn_geo_requirements.txt` contains the packages to install.

```
conda create -n acn_geo python=3.13.5
conda activate acn_geo
conda install --file path/to/acn_geo_requirements.txt
```

Folder Structure: to run properly this script, the following folder structure is required (if you already have the structure specified in Section 2.2, add only the commented folders):

```
climate_change_narratives/
 input/
   rawdata/                              # to be added
      gadm41_USA_shp/                    # to be added
          gadm41_USA_1.shp
          gadm41_USA_1.dbf
          gadm41_USA_1.shx
          ... (other shapefile components)
 output/
   data/
      processed_tweets/
        aggregated_data/
            tweets_full_clean_extracted.csv
            tweets_full_clean_extracted_analysis.csv
      geolocation/                        # to be added
          aggregated_data/                # to be added
          coordinates_nominatim/          # to be added
```

Before running the script, ensure that the GADM shapefile for US states is correctly placed and includes .shp, .shx, .dbf, .prj, etc.

Map: Input → Output: the input of this code is the output of Section 2.2, namely the processed tweets in .csv format. It takes as input both the random days dataset and the saturdays dataset. Moreover, it is important to ensure that the US shapefile is placed in the correct directory, as showed in the folder structure above.
Outputs is `output/data/processed_tweets/aggregated_data/analysis_tweets_geolocalized_usa.csv`, which is the US-only dataset with tweet and geolocation data merged, in .csv format.

3 OpenAI'Annotation Pipeline

This section explains how to query OpenAI's API to get annotated tweets. In this case, the goal is twofold. First, we are interested in assessing the relevance of each tweet to the discussion on climate change. Second, we are interested in identifying and labeling the charachters in each tweet as neutral, hero, victim, or villain.

3.1 04_gpt_annotation_tweets.py

This script sends tweets to OpenAI's Batch API for authomatic annotation. It uses a two stage approach: in stage 1 it filters tweets by relevance to the climate change topic, differenciating between

irrelevant tweets, tweets asserting climate change, tweets denying climate change, and relevant tweets for the public debate. In stage 2 it assigns narrative roles to the tweets labeled as highly relevant.

Required Software: prior to replicate this script run the following commands in Anaconda Prompt.

```
conda env create -f acn_data_manage.yml
conda activate acn_data_manage.yml
```

This allows to import the libraries to run the script (**here needs to be reviewed with Matteo**). Also ensure that your OpenAI API key is set as an environment variable: `set OPENAI_KEY = sk-...`

An additional requirement is that a helper file `functions.py` must be available and its folder must be added to `sys.path` in the python script:

```
module_directory = "path/to/functions.py"
sys.path.append(module_directory)
import functions
```

Folder Structure: to run properly this script, the following folder structure is required (if you already have the structure specified in Section 2.3, add only the commented folders):

```
climate_change_narratives/
 input/
   python/
     lookups/
   mTurk/tweets/
   openAI/                                # system messages for GPT stages
   rawdata/
      gadm41_USA_shp/

 output/
   data/
      original_tweets/
      processed_tweets/
      geolocation/
        aggregated_data/
        coordinates_nominatim/
      openAI/                            # annotation batches and outputs
          batch_input/narratives_tw/
          api_input/narratives_tw/
          api_output/narratives_tw/
          batch_id/narratives_tw/
          predictions/
```

Before running the script, ensure that the system messages (prompts) are correctly placed in `/input/openAI/`.

Map: Input → Output: the input of this code is the output of Section 2.3, namely the geolocated datasets. Other important inputs are the scripts to access OpenAI's API. These must be in JSON format and organized as follows:

```
{
  "SYSTEM_MESSAGE": "You are an average US citizen.
  The user will provide the content of a tweet posted from the US.
```

```
  Your task is to analyze the tweet within the context of
  US political discourse, particularly in relation to climate change.
  Respond in JSON format.
  ...
  Respond in JSON format, returning the value in the key \"r\"."
}
```

The final output is the full US tweets dataset with relevance scores and binary flags for narrative roles. This csv file is `output/data/openAI/predictions/predictions_gpt_full_2024-09.csv`

3.2 05_predictions_prep.py

This scripts merges the geolocated tweets and the two Visibrain datasets with the GPT predictions (output of Section 3.1. It derives analysis ready dummy columns for relevance and each narrative category.
Required Software: prior to replicate this script run the following commands in Anaconda Prompt.

```
conda env create -f acn_data_manage.yml
conda activate acn_data_manage.yml
```

As in Section 3.1, set the module directory to the path leading to `functions.py`.

Folder Structure: to run properly this script, the following folder structure is required (if you already have the structure specified in Section 3.1, add only the commented folders):

```
climate_change_narratives/
 input/
   openAI/
 output/
   data/
      processed_tweets/
        aggregated_data/
           analysis_tweets_geolocalized_usa.csv
      openAI/
        predictions/
           predictions_gpt_full_2024-09.csv
      openAI/
        aggregated_data/              # to be added
      dta/                            # to be added
```

Map: Input → Output:
the inputs for this script are the prediction dataset `predictions_gpt_full_2024-09.csv`, and the full geolocalized dataset from Section 2.3. Outputs consist in datasets in .dta format that will be used in the Stata scripts, namely `output/data/openAI/aggregated_data/df_x_pred_full_small.csv`. It creates also `output/data/openAI/aggregated_data/df_x_pred_full.csv` as a csv file, useful for the next script.

3.3 06_auxiliary_vars.py

This scripts enriches tweets with auxiliary text features for analysis. It compute counts (mentions, hashtags, caps, emojis), NRC emotions indicators, NLTK sentiment, spaCy entity counts, and text complexity measures. It saves this dataset in .dta format.
Required Software: prior to replicate this script run the following commands in Anaconda Prompt (if not done already).

```
conda env create -f acn_data_manage.yml
conda activate acn_data_manage.yml
```

As in Section 3.1, set the module directory to the path leading to `functions.py`.
Folder Structure: to run properly this script, the following folder structure is required (if you already have the structure specified in Section 3.1, add only the commented folders):

```
climate_change_narratives/
 input/
   rawdata/
      lookups/
         Emoji_Dict.p            # required
         Emoticon_Dict.p         # required
 output/
   data/
      openAI/
        aggregated_data/
           df_x_pred_full.csv    # INPUT to this script
      text_features/
        x_pred/                  # will store chunk pickles
      dta/                       # Stata exports (.dta)
```

Map: Input → Output:
the inputs for this script `output/data/openAI/aggregated_data/df_x_pred_full.csv`, and the full geolocalized dataset from Section 2.3. Moreover, it's important to get the emoji and emoticon dictionaries in the correct directories. Main output is a stata .dta with auxiliary features for each tweet, namely `output/data/dta/df_x_auxiliary_vars.dta`. It also creates a dataset for the experimental analysis of the paper, that is `output/data/dta/df_x_auxiliary_vars_experiment.dta`.

3.4 07_gpt_annotation_profiles.py

This script uses the OpenAI Batch API to annotate user profile descriptions (from Twitter authors). It loads all user descriptions from the geolocated dataset, drops duplicates and empties, and chunks them into batches. It submits each batch to GPT for annotation with a system message, labeling profiles on politics, religion, education, and children. Lastly, it saves batch outputs as CSVs, then merges them into a final annotated dataset.

Required Software: prior to replicate this script run the same commands as Section 3.1 in Anaconda Prompt. Also ensure that your OpenAI API key is set as an environment variable: `set OPENAI_KEY = sk-....` Lastly, as in Section 3.1, set the module directory to the path leading to `functions.py`.

Folder Structure: to run properly this script, the following folder structure is required (if you already have the structure specified in Section 2.3, add only the commented folders):

```
climate_change_narratives/
 input/
   openAI/
      system_message_profile.json              # the message for OpenAI

 output/
   data/
      processed_tweets/
        aggregated_data/
           analysis_tweets_geolocalized_usa.csv
      openAI/
        batch_input/profiles/
        api_input/profiles/
```

```
        api_output/profiles/
        batch_id/profiles/
        predictions/
           profiles_gpt_full_2024-10.csv        # final output in .csv
      dta/
         df_x_profiles.dta                      # Stata export
```

Before running the script, ensure that the system messages (prompts) are correctly placed in `/input/openAI/`.

Map: Input → Output: the input of this code is the output of Section 2.3, namely the geolocated dataset. Other important inputs are the scripts to access OpenAI's API. These must be in JSON format and organized as follows:

```
{
  "SYSTEM_MESSAGE": "You are tasked with analyzing the Twitter
  profile description of a U.S. citizen. The user will provide
  the profile description text. Your task is to classify the
  profile based solely on the information explicitly stated in
  the description. Respond in JSON format using the following
  categories:

    ...

  Final Output:
  Respond in JSON format containing the following keys:
  - 'a': ('1','2','0') for political leaning
  - 'b': ('1','0') for religiosity
  - 'c': ('1','0') for level of education
  - 'd': ('1','0') for has children."
}
```

The final output is a .csv file with annotation as described in the JSON message. The same dataset in .dta format.

3.5 08_experiment_classification.py

This script uses the OpenAI Batch API to annotate open-ended memory recall answers from different experiments (HH, HHV, VVH). It takes free-text responses from .dta files, splits them into manageable chunks, and sends them to GPT for classification. GPT outputs structured labels (e.g., mentions of green technologies, fossil industry, etc.). The script processes both main experiments and their follow-ups, merges the GPT annotations back to the participant IDs, and saves the results as Stata .dta datasets.

Required Software: prior to replicate this script run the same commands as Section 3.1 in Anaconda Prompt. Also ensure that your OpenAI API key is set as an environment variable: `set OPENAI_KEY = sk-...`. Lastly, as in Section 3.1, set the module directory to the path leading to `functions.py`.

Folder Structure: to run properly this script, the following folder structure is required (if you already have the structure specified in Section 2.3, add only the commented folders):

```
climate_change_narratives/
 input/
   python/
     lookups/
        Emoji_Dict.p
```

```
        Emoticon_Dict.p
    openAI/
       system_message_memory_test2.json
       system_message_memory_test3.json
       system_message_memory_test4.json

 output/
    data/
       dta/
         experiment_hh.dta         #
         follow_up_hh.dta         #
         experiment_hhv.dta         #
         follow_up_hhv.dta         #
         experiment_vvh.dta         #
         follow_up_vvh.dta         #
```

Before running the script, ensure that the system messages (prompts) are correctly placed in `/input/openAI/`.

Map: Input → Output: the input of this code is the output of the Stata code for the data preparation, namely the .dta files marked in the folder structure showed above. Other important inputs are the scripts to access OpenAI's API. These must be in JSON format and organized as follows:

```
{
  "SYSTEM_MESSAGE": "You are a researcher tasked with analyzing
  responses to an open-ended survey question. Your task is to
  identify features of the text as per the criteria below. Respond
  strictly in JSON format.

  1. Character Analysis:
     - Identify whether the response mentions specific

     ...


  4. Final Output:
     Respond in JSON format with keys:
       - "green_tech": 0{4
       - "people_us": 0{4
       - "causal_links": 0{1

  Note: Ensure the JSON strictly follows this structure.
  Provide no extra explanations or commentary."
}
```

The final output is a .dta file with annotation as described in the JSON message.