Web-Scraped Tweet Timemap Analysis

Paola Gonzalez

November 2024

1 Project Description, Background, and Motivation

Project Description: This project focuses on the collection and analysis of web-linked resources (URIs) shared through tweets, specifically around social movements. By scraping tweets that discuss key topics related to movements such as climate change, racial justice, and gender equality, this project aims to examine how well the linked content (often news articles, studies, or other online resources) is preserved over time. The study specifically tracks how these web resources are archived using web mementos (time snapshots), assessing the availability and longevity of these crucial records in the digital archive. The code and data related to this project can be found at `https://github.com/paolagonzalez0/web-scraping-project`.

Background: Social movements often generate significant discourse and media coverage on social platforms like Twitter. With the rapid evolution of technology and the speed at which digital content is created, much of the information shared online can easily be lost, altered, or forgotten over time. The preservation of this information is critical not only for historical analysis but also for legal and social purposes. As technology advances, there is a growing need for the systematic archiving of web content to ensure that digital heritage is maintained. Organizations, including governmental bodies, educational institutions, and private entities, have growing obligations to ensure that their digital records are kept accessible for future generations, for use in research, policy-making, and legal contexts.

Motivation: The motivation behind this project lies in the increasing importance of digital preservation for societal understanding. With the vast amount of data being generated, it is crucial to assess how effectively online content is preserved for future reference. By analyzing the longevity of linked content within tweets related to social movements, this project seeks to understand how well past online discussions are retained. The potential loss of archived web materials could lead to gaps in understanding social change, political shifts, and technological impacts on society. In particular, archived web materials serve

essential functions in areas such as historical research, legal proceedings, and cultural preservation. This project aims to contribute to the broader effort to safeguard our digital history and ensure that pivotal social discussions remain accessible in the future.

2 Data Description

Data Collection: A central component of this project involved the collection of data from Twitter, specifically through web scraping techniques using a headless browser. The scraping functionality was implemented to accept specific keywords as input and retrieve relevant tweets that appear on the search engine results page. This project specifically focused on tweets associated with various social movements, aiming to capture public discourse and related resources linked within these tweets. The data scraping process was structured and managed through a set of Python scripts stored in the following files: `collecting_tweets.py`, `scrape_twitter.py`, `process_tweets.py` and `util.py`.

Data Organization: All collected data is stored within the `scraped_tweets` directory. Each data collection session produces a compressed JSONL file, where each line represents an individual tweet in JSON format. Each entry in the JSONL file includes key information such as the raw tweet text, the number of likes, and any associated links embedded within the tweet. This structured format enables efficient parsing and analysis, providing a straightforward structure for assessing the content and engagement metrics associated with each tweet. A sample tweet in JSON format is available in the `example.json` file, located within the documentation directory.

3 Project Summary and Objectives

This project is structured into four key phases:

1. **Tweet Collection:** Gather tweets related to social movements through web scraping.
2. **URI Extraction:** Identify and extract URIs embedded within the collected tweets.
3. **Timemap Retrieval:** Use MemGator to obtain timemaps (lists of mementos) for each extracted URI.
4. **Memento Analysis:** Analyze the mementos to uncover patterns in the preservation of digital resources associated with social movements.

4 Progress and Next Steps

Significant advancements have been made in the tweet collection phase of this project, particularly in developing and implementing web scraping code to gather a representative sample of tweets. The core data collection functionality is stored in the script `collecting_tweets.py`. This script includes automated, well-documented functions designed to improve readability and usability. Supporting backend functionality is modularized across additional scripts for better organization and scalability. These scripts include `scrape_twitter.py`, `process_tweets.py`, and `util.py`.

The URI extraction phase has also been completed with the associated code stored in `extracting_links.py`. This script iterates through all scraped tweets stored in the `scraped_tweets` directory, extracts URIs, and stores them into the `tweet_links.txt` file.

To streamline the workflow, I have created a `main.py` script to allow users to run the entire program seamlessly and to present a clean and professional interface. Currently, the main function is written to complete tweet collection and URI extraction. As I make more progress, I will continue to add the remaining functionality to the project.

The next phase involves extracting timemaps for each URI. To accomplish this, I am currently developing a script to deploy MemGator, a Memento Aggregator CLI, as a Docker container. This process will generate comprehensive timemaps for each link, detailing web archiving snapshots over time. Given the computational intensity of this stage, initial efforts will focus on testing and retrieving timemaps for a small sample of data. This approach will allow for optimization and troubleshooting before scaling to the full dataset.

5 Prerequisites

Prior to execution, ensure the following prerequisites are fulfilled for the program to function correctly.

5.1 Clone the repository

Navigate to your desired local directory and execute the following command in your terminal.

```
git clone https://github.com/paolagonzalez0/web-scraping-project.git
```

5.2 Create a Twitter account

To complete tweet collection, a Twitter account is needed. To set up an account, follow the instructions provided here. Note: To avoid any disruptions, do not use your personal account or an account you can't afford to lose. There's a possibility that the account might be flagged or suspended for suspicious activity

during the scraping process. Consider creating a dedicated account specifically for this project.

5.3 Download Docker Desktop

Download and install Docker Desktop here. This tool will be used for retrieving the URI timemaps in phase three of the project. For timemap retrieval to execute properly, pull a published image from a Docker image registry to run Memgator by running the following command in your terminal:

```
docker image pull oduwsdl/memgator
```

5.4 Install necessary libraries and tools

5.4.1 Python Version

The program requires Python 3.8 or newer. Ensure Python on your device is up to date by running the following command in your terminal:

```
python --version
```

5.4.2 Required Libraries

Install dependencies listed in `requirements.txt` by running the following command in your terminal:

```
pip install -r requirements.txt
```

5.4.3 Browser Automation Tools

Ensure Playwright browser tool is installed by running the following command in your terminal:

```
playwright install
```

5.4.4 Network Access

Ensure you have stable internet access for scraping.

6 Project Execution

After meeting the prerequisites, the program can be run by navigating to the scripts directory and running the following command in terminal:

```
python main.py
```

Note: This program has been run previously to collect sample data. By running the program again, new data will be added to the previously existing data.

References

[1] S. Alam, M.L. Nelson. *MemGator*, https://github.com/oduwsdl/MemGator

[2] A. Nwala. *Homework 2 - Archiving the Web*, https://github.com/anwala/teaching-web-science/tree/main/fall-2024/homework/hw2

[3] Docker. *Docker Desktop* https://www.docker.com/products/docker-desktop/