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 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.

3 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. The tweet scraping process was structured and managed through a set of Python scripts.

Data Augmentation: This project aims to capture public discourse and related resources linked within these tweets. Hence, the URIs from collected tweets are extracted, filtered, and stored in a text file. Following this, the timemap of each URI is collected and stored to be used for analysis.

Data Organization:

- `documentation`: This directory contains the project documentation and supporting example files.
- `scraped_tweets`: Sample tweet data is stored within this 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 URIs embedded within the tweet. A sample tweet in JSON format is available in the `example.json` file, located within the documentation directory.

- `scripts`: This directory contains all the back-end scripts and code required for the execution of the project.
  - `analysis.py`: Contains code used for the analysis of URI timemaps and mementos.
  - `collecting_tweets.py`: Contains code to scrape and save tweets for a given list of keywords.
  - `extracting_links.py`: Contains code to extract and filter URIs in raw tweet data.
  - `get_timemaps.py`: Contains code to fetch the timemaps for the given URIs.
  - `main.py`: Contains code to run the tweet collection, URI extraction, and timemap retrieval processes.
  - `process_tweets.py`: Contains code to process tweet data in tweet collection process.
  - `scrape_twitter.py`: Contains code to execute the Twitter scraping process.
  - `util.py`: Contains code used in tweet collection process.
- `timemaps`: The sample timemap data is stored within this directory. Each file name corresponds to its URI based on its line number in the `tweet_links` text file. One file contains a JSON object holding the original URI, a list of mementos, and the earliest and latest memento captured for the corresponding URI.
- `analysis.ipynb`: Runs and showcases project results.
- `requirements.txt`: This file contains the following libraries and dependencies required for the execution of the project.
- `social_movements.txt`: This file holds a list of various social movements used in tweet collection, where one line in the file represents one social movement.
- `tweet_links.txt`: The extracted URIs are saved in this file, with each line in the file representing a single URI.

4 Project Execution

This project can be executed in two approaches. The first approach involves both data collection and analysis, covering all four phases of the project: tweet collection, URI extraction, timemap retrieval, and memento analysis (4.1). The second approach uses previously existing data to run phase four of the project exclusively, without collecting new data (4.2).

4.1 To execute all four phases of the project:

- Ensure the prerequisites in next section of the documentation are met.
- Run the tweet collection, URI extraction, and timemap retrieval processes by navigating to the scripts directory and running the following command in terminal:

  ```
  python main.py --total_links_to_scrape 150
  ```

  - Upon running this command, the Nightly browser should open automatically to the Twitter log in page. From here, input your login credentials from your new Twitter account and the program will begin executing the first three phases of the project.
  - In this example, the program will collect data until 150 URIs have been saved. To collect more or less URIs, replace 150 with the desired number of URIs to be analyzed.
  - 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.
- Next, the analysis of the project can be executed by opening and running the code in `analysis.ipynb`. This notebook showcases the results and summary statistics calculated on the collected data.

4.2 To execute the project without data collection:

- The analysis of the project can be executed by opening and running the code in `analysis.ipynb`. This notebook uses previously collected data to calculate the results and summary statistics showcased in final analysis.

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 Methods

The main program begins by randomly selecting five social movements and performing an initial iteration of the tweet collection process, saving 200 tweets in a file within a specified directory. Next, all URIs are extracted from the tweet files. If the program collects a sufficient number of URIs, as specified by the user, it proceeds to the timemap retrieval process. Otherwise, it continues selecting new social movements, collecting and creating additional tweet files, and extracting more URIs until the target is reached. It is important to note that the timemap retrieval process will only commence once the specified URI count is achieved. The extracted URIs are stored in a text file, which is then used to retrieve timemaps for all URIs. The timemaps are retrieved by using Memgator, a Memento Aggregator CLI, as a Docker container. As each timemap is retrieved, it is saved as a JSON file in a specified directory. Once the timemaps are collected, the main program completes execution. At this stage, a Python notebook can be run to perform a memento analysis on the gathered data. Below is a more detailed explanation of each phase of the project.

6.1 Tweet Collection

The tweet collection process begins by picking `n` random topics from a list to focus on for collecting tweets. Next, the program iteratively scrapes tweets for each selected topic, storing them in separate `gzip` files. After collecting tweets, the files are combined, and the total number of tweets is checked. If not enough tweets are collected, additional tweets are collected until the target is met. The result is a consolidated file containing all the tweets for the selected topics, which is used for further processing.

6.2 URI Extraction

The URI extraction process works by initializing a text file to collect the URIs, if it does not exist. The program iteratively opens each tweet collection file and parses each tweet element to extract URIs from the JSON object, if they exist. It filters out irrelevant URIs, like retweets or those linking to video content. If the link is relevant and unique, it writes it into the text file. When the target number of links is collected, the script stops URI extraction.

6.3 Timemap Retrieval

The timemap retrieval process works by requesting and saving timemaps for given URIs using the MemGator tool. The program runs a Docker command to fetch timemap data in JSON format, extracts the valid JSON portion. It also includes error handling to manage timeouts or command failures. The final function creates a directory to store the results and iterates through the list to collect and save each timemap as a uniquely named JSON file in a specified directory.

6.4 Memento Analysis

This code performs memento analysis for the URI timemap data. Some of the main features include counting the mementos for each URI, calculating the age of the earliest memento for each URI, and creating a memento distribution table. This script also provides summary statistics including the percentage of archived URIs and the oldest URI and its first memento date. It compiles all features into a dataframe and provides a visualization to showcase the correlation between URI ages and their number of mementos, allowing for a comprehensive analysis of the archiving status of each URI. These functions are all run in a Python notebook.

7 Results

The main results of this project can be found in the `analysis.ipynb` file. However, keeping in mind the project motivation, what do these results reveal about the preservation of digital resources related to social movements? Similar to other online resources, URIs associated with social movements are poorly preserved. The analysis indicates that only about 55% of the collected URIs were archived. This highlights a significant gap in preserving digital materials, particularly those that drive or support social change. With approximately 45% of potentially valuable data left unarchived, there is a substantial risk of losing critical information due to the absence of effective preservation strategies.

Along with this, the quality of preservation for the archived data is also concerning. The oldest memento identified dates back to September 2000, showing limited historical coverage. Additionally, about 12% of the URIs had their first memento archived during the same week the data was collected, indicating these resources were not preserved until very recently. Even for URIs with at least one memento, there remains a notable gap in their preservation.

This analysis highlights the widespread issue of data loss, not only for resources tied to social movements but for all digital content across the web. It serves as a reminder of how much information we are at risk of losing, emphasizing the need for improved digital preservation practices to conserve these resources for future generations.

8 Future Work

With the data collected, the program was able to do a good amount of analysis and provide meaningful results. However, there are a couple of areas of the project that could be improved to enhance the results.

- **URI Filtering:** The current URI filtering process is straightforward, relying on a static list of hardcoded patterns to identify unimportant links.

While this approach removes some irrelevant links, it is not the most comprehensive approach, since some unneeded links may still pass through into the dataset. To enhance this process, incorporating a basic classification model in the future could significantly improve link filtering. By employing a more intelligent approach, the filtering would ensure that only relevant links are included, ultimately leading to more accurate and meaningful analysis.

- **Analysis:** The majority of this project focused on data collection, which meant much of the data used for analysis came from data augmentation. As a result, feature engineering was limited, and the analysis was less complex than initially intended. Allocating more time to feature engineering would allow for a more detailed and comprehensive analysis, enhancing the overall depth and quality of the findings.

By improving these elements of the program, the analysis and results could be more detailed and give more information, despite the lack of web and URI preservation.

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/

[4] Argparse Tutorial. *Python* https://docs.python.org/3/howto/argparse.html