Computational Analysis and Visualization of Twitter Activity During the Mati Wildfire: An Advanced Python Programming Approach

Dionisios N. Sotiropoulos
Advanced Python Programming

January 22, 2025

Dataset Description

The dataset `https://www.dropbox.com/scl/fi/y8wvktb5lefnozak13aru/mati.csv?rlkey=hl3f7wpwe6ruadgx4v0cecya0&st=uxf341rz&dl=0` under analysis comprises Twitter data related to the tragic Mati Fire, which occurred in the Attica region of Greece in July 2018. This catastrophic wildfire was one of the deadliest in modern European history, causing significant loss of life, destruction of property, and widespread environmental damage. Social media platforms, particularly Twitter, played a pivotal role during this event, serving as a medium for information dissemination, public reactions, emergency communications, and the mobilization of relief efforts.

The dataset consists of tweets collected over a specified time period and includes information such as:

- **Author Information:** Anonymized identifiers for the authors of the tweets, enabling analysis of user-level activity.
- **Timestamp:** The exact time and date when each tweet was posted, facilitating temporal analysis and identification of activity bursts.
- **Geolocation Data:** If available, the geographic origin of the tweets, useful for spatial analysis and identifying affected regions or clusters of activity.
- **Tweet Text:** The content of the tweets, in the Greek language, which serves as the primary source for textual and sentiment analysis.
- **Engagement Metrics:** Metrics such as likes, retweets, and replies, providing insight into the public response and virality of specific tweets.
- **Retweet Indicators:** Markers to distinguish original tweets from retweets, which are essential for understanding content propagation and author influence.

Assignment Questions

This section outlines the challenging computational tasks you are expected to complete. Beyond implementing the required computations, you are encouraged to critically evaluate the dataset, justify your methodological choices, and clearly articulate the assumptions underlying your analysis. Each solution should demonstrate thoughtful reasoning and provide insights into the implications of your results, ensuring a comprehensive and well-substantiated approach.

1. Filtering Irrelevant Tweets

- Develop a computational mechanism to identify tweets that are unrelated to the natural catastrophe in Mati by analyzing their text content.
- Keywords and pattern-matching techniques should be used to classify tweets, while incorporating manual inspection of samples to refine filtering rules.

2. Temporal Burst Detection with Moving Averages

- Compute the daily tweet volume over the dataset timeline.
- Apply a rolling average to smooth the data, allowing better detection of spikes.
- Identify burst periods by comparing the daily tweet volume against the rolling average and flagging days where the volume exceeds a threshold (e.g., 2 standard deviations above the rolling mean).
- Visualize these bursts with a time-series plot, clearly highlighting detected periods.

3. Author Activity Ranking

- Group by `author_id` and compute the following metrics:
  - Total tweets posted by each author.
  - Time span of activity for each author (difference between their first and last tweet timestamps).
  - Average time interval between consecutive tweets.
- Use a weighted scoring system to rank authors by their activity level based on the computed metrics.
- Provide visualizations or tables to showcase the most active users.

4. Time-of-Day Analysis

- Extract the hour from each timestamp in the `timestamps` column.
- Group by hour and compute the average tweet volume for each hour of the day.
- Create a bar plot visualizing tweet activity by hour.
- Identify time periods with unusually high or low activity, providing insights into user behavior patterns.

5. Engagement Metrics Aggregation

- Group tweets by `author_id` and compute aggregate engagement metrics:
  - Total counts for `like_count`, `retweet_count`, and `reply_count`.
  - Engagement rate per tweet, calculated as the sum of engagement metrics divided by the total number of tweets for each author.
- Rank authors based on their average engagement rate and highlight the top contributors to the dataset's overall engagement.

6. Time-Gap Analysis Between Tweets

- Calculate the time difference between consecutive tweets for each `author_id`.
- Group by `author_id` and compute:
  - Average time gap between consecutive tweets.
  - Standard deviation of time gaps.
  - The number of tweets with unusually short gaps (e.g., less than 1 minute) or unusually long gaps (e.g., more than 1 day).
- Identify authors with irregular tweeting patterns and summarize their behavior.

7. Weekly Activity Patterns

- Extract the day of the week and the week of the year from the `timestamps` column.
- Group by the week of the year and compute the total tweet volume for each week.
- Create a heatmap where rows represent weeks and columns represent days of the week, with cell values corresponding to tweet volume.
- Analyze the heatmap to identify weekly trends or anomalies in tweet activity.

8. Author Retweet Dependency

- Identify retweets by analyzing the `text` column or retweet indicators (`RT`).
- Group by `author_id` and compute:
  - The percentage of tweets that are retweets for each author.
  - The retweet-to-original content ratio.
- Rank authors based on their reliance on retweets versus original content, and visualize the distribution of retweet dependency across users.

9. Burst-Origin Analysis

- Use the burst periods identified in Task 2 to analyze user contributions to these bursts.
- For each burst, identify the most active authors during the preceding time window (e.g., 6 hours before the burst onset).
- Group by `author_id` and compute:
  - The percentage contribution of each author to the burst (tweets during the burst vs. all tweets).
  - The lag time between the author's first tweet in the time window and the burst onset.
- Create a ranked DataFrame highlighting authors who were key drivers of activity during bursts.

10. Identification of Potentially Influential or Manipulative Authors

- Leverage the results from previous tasks to analyze the potential influence of specific authors on tweet volume bursts.
- Answer the following sub-questions to systematically address this task:
  - Which authors consistently contribute significantly to bursts across multiple periods? Calculate their average contribution percentages during bursts.
  - Are there authors whose tweets appear shortly before a burst begins, possibly acting as catalysts? Use lag time metrics from Task 9 to identify such patterns.
  - Analyze retweet patterns from Task 8. Are there authors whose content is disproportionately retweeted during bursts, suggesting a role in propagating information?
  - Are there any patterns in the timing or frequency of tweets from these authors that indicate coordinated activity or attempts to influence public perception?

- Provide visualizations, such as contribution percentage plots, retweet frequency charts, or activity timelines, to support your analysis.
- Critically evaluate your findings and discuss whether the observed patterns indicate genuine influence, coordinated activity, or other possible explanations.

Deliverables

Students are required to submit a comprehensive report documenting their solutions to the assignment tasks. The report must include clear and concise explanations of the methodologies used, supported by appropriate visualizations (e.g., histograms, line plots, heat-maps) to illustrate findings effectively. All visualizations should be well-labeled, with descriptive titles, axes, and legends where applicable. Students must also provide the complete Python code for each task, ensuring that the code is well-documented with comments explaining key operations and design choices. Additionally, include execution outputs such as screenshots of results or log outputs to validate the correctness of the code. The final submission should be organized, professional, and presented in a logical order, enabling readers to follow the problem-solving process seamlessly. Code must be submitted in a runnable format, and any external libraries or dependencies should be clearly stated in a requirements.txt file. **You may work in groups of 3 students at maximum**.