BuzzFace

All files and scripts included were created by the authors, except for `facebook-fact-check.csv`, which is the basis for the dataset. This file was created by a team at BuzzFeed as a result of the study detailed in: Hyperpartisan Facebook Pages Are Publishing False And Misleading Information At An Alarming Rate. The authors took the news posts analyzed in the this BuzzFeed dataset and created the included tools to obtain Facebook comments and metadata related to each, along with the actual content of the text articles themselves. Also provided are tools to organize this data and easily manipulate it.

Structure

The files included are as follows:

- `ANONYMIZED_COMMENTS.json` - A dictionary where the keys are the comment IDs of the Facebook comments of the posts in question, and the values are the Ego-IDs of the users that made those comments. Used in the API.
- `collectPosts.py` - A Python script which goes through all the Facebook posts in the BuzzFeed set and obtains their metadata, comments, and attachments. Dumps this data into `posts.json`, `comments.json`, and `attach.json` files, respectively.
- `facebook-fact-check.csv` - The basis for our dataset. Created by BuzzFeed, this is a CSV which provides data concerning all the posts made by 9 news outlets over a 10-day period in September 2016. The provided fields are: account id, post id, category, page, post URL, date published, post type, rating, debate, share count, reaction count, comment count.
- `get_post_times.py` - A Python script which creates the `POST_TIMES.json` file, which provides the API and user with a record of when the posts in question were first made on Facebook.
- `helper.sh` - A bash script which simply runs all the Python scripts in the correct order, with the appropriate arguments.
- `install.sh` - A bash script which installs the dataset. Should be run using sudo!
- `order_by_thread.py` - A Python script which takes all the Facebook comments obtained and dumps them into `THREADS.json`, which is organized by thread.

- `order_by_user.py` - A Python script which takes all the Facebook comments obtained and dumps them into `USERS.json`, which is organized by user.
- `scrape.py` - A Python script which obtains the content of the text articles themselves, and dumps the results into the various `scraped.json` files.
- `threaded_comments.py` - A Python script which downloads the full threaded Facebook commentary structure. Normal Graph API access just returns the top-level comments, this file is needed to get the replies as well.
- `main.py` - A Python script which allows the user to use the built-in API for the data.
- `users.py` - A Python script which implements the User class in the built-in API.
- `threads.py` - A Python script which implements the Thread class in the built-in API.

Installation

Use the `install.sh` script to download and organize the dataset, using the following command in a shell:

```
sudo bash install.sh
```

Input password if asked to. This will create a directory called `dataset` which has a directory for each outlet. Inside each outlet's directory, each post is given its own directory containing the following files:

- `comments.json` - the top-level Facebook comment data as returned by the Graph API
- `replies.json` - the full Facebook comment data as returned by the Graph API
- `posts.json` - metadata about the post as returned by the Graph API
- `scraped.json` - if the post was a text article, this is the result from accessing the article itself
- `attach.json` - metadata about any attachments made to the post as returned by the Graph API.

Additionally, `THREADS.json` and `USERS.json` which are files which consist of all the Facebook comments organized by thread or user, respectively. A file called `POST_TIMES.json` is created during this process which is a dictionary that has keys post IDs and values the date and time these posts were first created. These files facilitate use of the API, which is further discussed below. These comments will have the additional randomized anonymous Ego-IDs we created, in order to preserve the ability to study the data by user. If the user would like to look at the Ego-IDs of comments in the `comments.json` or `replies.json` files, the provided file `ANONYMIZED_COMMENTS.json` is a dict where the keys are comment IDs, and values are the associated Ego-IDs. These Ego-IDs are based on the comments as they stood when we first accessed the data nearly a year ago. There are sure to have been additional comments made since then. These will still appear in the dataset, but their Ego-ID will all be simply "NA". It's important for the researcher to keep this in mind if they intend to user `users.py` for user analysis. The "NA" category can be handled however the researcher likes, but it may be best to ignore it.

Since this process takes quite a long time, the script makes use of the nohup command, which will force the script to continue working even in the background even if the terminal session is closed. If you'd rather not run the process in the background, execute the script instead with the following command:

```
sudo bash helper.sh FB_ID FB_SECRET DISQUS_ID DISQUS_SECRET
```

where `FB_ID`, `FB_SECRET`, `DISQUS_ID`, `DISQUS_SECRET` are your API keys.

As alluded to in the code, it's possible to cut down on the time it takes to acquire the data by changing the wait time in between API calls. We reiterate the user may do this at their own risk of suprassing Facebook rate limits.

Far more detailed descriptions of the specifics of the dataset and meanings of the various keys and values, along with the API can be found in the paper associated with this dataset.

API Features

Using main.py

This script is for the user to access the data. Almost everything you'd want to to do has a function. Anytime a function has a keyword argument of choice, the user may choose to filter the data by:

- choice = "All" is default, returns all the data
- choice = "Top" gives back the top-level comments
- choice = "Reply" gives back the replies

The functions the user will want to use are:

Threads

- threadText(threadID, choice) - self-explanatory
- threadTimes(threadID, choice) - self-explanatory
- threadUsers(threadID, choice) - self-explanatory

- threadCommentTime(threadID, choice) - returns a list of tuples of the comments with post-times
- threadUserCounter(threadID, choice) - returns a Counter of the users that commented in the thread
- cutThread(threadID, date_time) - allows the user to look at a thread up until a certain time. the date_time input must be a datetime object, which is of the form: datetime.datetime(year, month, day, hour, minute, second).
- threadResponse(threadID, choice) - returns response times for the entire thread

Users

- userText(userID, choice) - analogous to above
- userTimes(userID, choice) - analogous to above
- userThreads(userID, choice) - analogous to above
- userTextTimes(userID, choice) - analogous to above
- userThreadCounter(userID, choice) - analogous to above
- userResponse(userID, choice) - returns response times for the user

It's important to note that after making a cut Thread, the user must investigate the features of the thread manually by accessing the class parameters. There aren't functions built to handle cut threads.

Class parameters

Threads

- THREAD - all comment data
- threadID
- post_time - original post time
- all_text
- all_times
- all_text_time - zip of all_text and all_times
- all_users
- top_text
- top_times
- top_text_times
- top_users
- reply_text
- reply_times
- reply_text_time
- reply_users

Users

- COMMENTS - all comment data
- userID
- all_text
- all_times
- all_text_time - zip of all_text and all_times
- all_threads
- top_text
- top_times
- top_text_times
- top_threads
- reply_text
- reply_times
- reply_text_time
- reply_threads

Contact

- Giovanni C. Santia - gs495@drexel.edu
- Jake Ryland Williams - jw3477@drexel.edu

Department of Information Science

College of Computing and Informatics

Drexel University

30 N. 33rd St.

Philadelphia, Pennsylvania 19104

Citation

The paper associated with and describing this data set is: *BuzzFace: a News Veracity Dataset with Facebook User Commentary and Egos*, found in ICWSM-18.