This document summarizes the approach which CEDR Digital Corps has used to 'auto-mine' Twitter.

Objective

Identify tweets that contain information relevant to disaster response topics. For example, reports about shelters opening or closing, or photographs posted of the damaged caused by a disaster.

Search Approach

First, we identify keywords that define a topic; for shelters, that can be as simple as "(shelter OR FEMA)".

Next, we combine the topic search with 3 other approaches to narrow down the results and ensure they are actually relevant to incident we are focused on:

1. Geosearch. Using Twitter's geosearch, when looking for some tweets like damage photographs, a broad geosearch is sufficient, using coordinates central to the incident and a broad radius of 200km or other appropriate distance. For certain topics like shelters that are explicitly about a particular location, we can do even better if we have the latitude / longitude of those locations. As CEDR often maintains a master list and ArcGIS map layer of shelters during an incident, we can use a 'brute force' approach, creating searches which include several shelter locations combined with the topic keywords, resulting in something like:

(shelter OR FEMA) AND ((geocode:36.81,-76.73,1km) OR (geocode:36.94,-76.59,1km) OR (geocode:35.06,-78.92,1km) OR (geocode:34.82,-78.24,1km) OR (geocode:34.94,-78.28,1km) OR (geocode:34.97,-78.33,1km) OR (geocode:34.99,-78.5,1km) OR (geocode:35.17,-78.47,1km) OR (geocode:35.17,-78.32,1km) OR (geocode:37.12,-76.43,1km) OR (geocode:37.67,-75.67,1km) OR (geocode:37.77,-75.64,1km) OR (geocode:37.92,-75.55,1km) OR (geocode:37.67,-75.72,1km) OR (geocode:37.68,-75.72,1km) AND -filter:nativeretweets

2. Unfortunately, many people posting will not have turned on geolocation for Twitter, and so the geosearch approach will miss their tweets. So the next option is combing the topic keywords with the incident name / hashtags. ("#Florence" OR "Florence"). For previous, smaller scale incidents like Hurricane Lane, that worked. But in larger scale disasters like Hurricane Florence and Hurricane Michael, the volume for of tweets generated has been too great, and this approach becomes unworkable.

3. Place names. Another brute force approach, this time using the town/location names of places in the affected area. CEDR first tried this technique in Hawaii during Hurricane Lane, where it worked wonderfully because Hawaii has such unique names. But it still works surprisingly well even in the mainland US areas affected by storms such as Florence & Michael. So that can give a search like:

(shelter OR FEMA) AND AND (neuse OR kershaw OR whitaker OR davis OR jarrett OR marshallberg OR glouchester OR harkers OR beaufort OR Morehead OR "atlantic beach" OR "pine knoll" OR "cedar point" OR swansboro OR carteret OR hammocks OR sneads OR chadwick OR topsail OR jacksonville OR "frying pan" OR wrightsville OR myrtle OR "new bern" OR Wilmington OR Moorehead) AND -filter:nativeretweets

Automation Approach

Old Method

So those are the searches, and then comes the automation, which is primarily driven by a tool called Zapier (https://zapier.com/apps/) which has a direct Twitter API integration. We create a Zapier "zap" routine for each Twitter search. Zapier then runs the search and puts the results in a Google Sheet on the appropriate tab for the topic (i.e. the Shelters tab). Then a separate Zapier routine periodically checks the tab and copies the new entries to CEDR's Slack instance, posting them to a dedicated channel that our team watches and then manually reviews for appropriate action. (This two-step process allows us to have multiple searches/zaps for a topic all feed into the tab & get combined / de-duplicated, and then get moved as an integrated bunch to Slack for action).

The fully Zapier-driven approach works, but it's not easily updated. Adding new locations requires manually create new Zaps any time I want to add place names / geolocations.

So that led to a new approach...

New Method

In our new automation method, we use Blockspring's Google Sheets add-on (https://www.blockspring.com/blog/blockspring-for-google-sheets) as the primary engine driving our Twitter searches.

Blockspring has an integration to Twitter, and provides a way for us to set up a Google Sheet with all of our searches in a way that can be more easily updated. This Google Sheet gathers tweets with damage photographs from locations in Florida panhandle impacted by Hurricane Michael:

https://docs.google.com/spreadsheets/d/11h-ytwomh0kpzzl13aui4iOLjXPbLq1RLkwI8hlhb5Y/edit?usp=sharing

First, look at the Places tab, and you'll find that it is exactly what it sounds like: a list of places. In this case, cities and towns within the counties of Florida we know were affected by Hurricane Michael.

Next, go left to the Keywords tab, and you'll see how the places get assesmbled together in groups of ten into partial queries, like the one in cell C11:

(Callaway OR "Cedar Grove" OR "Laguna Beach" OR "Lower Grand Lagoon" OR "Lynn Haven" OR "Mexico Beach" OR "Panama City" OR "Panama City Beach" OR Parker OR "Pretty Bayou")

Finally on the Searches tab, the groups of locations get joined with whatever topic keywords search has been defined (pulled from cell F2 of the Searches sheet), such as:

("flooding" OR "flooded" OR "florence" OR "damage" OR "recovery")

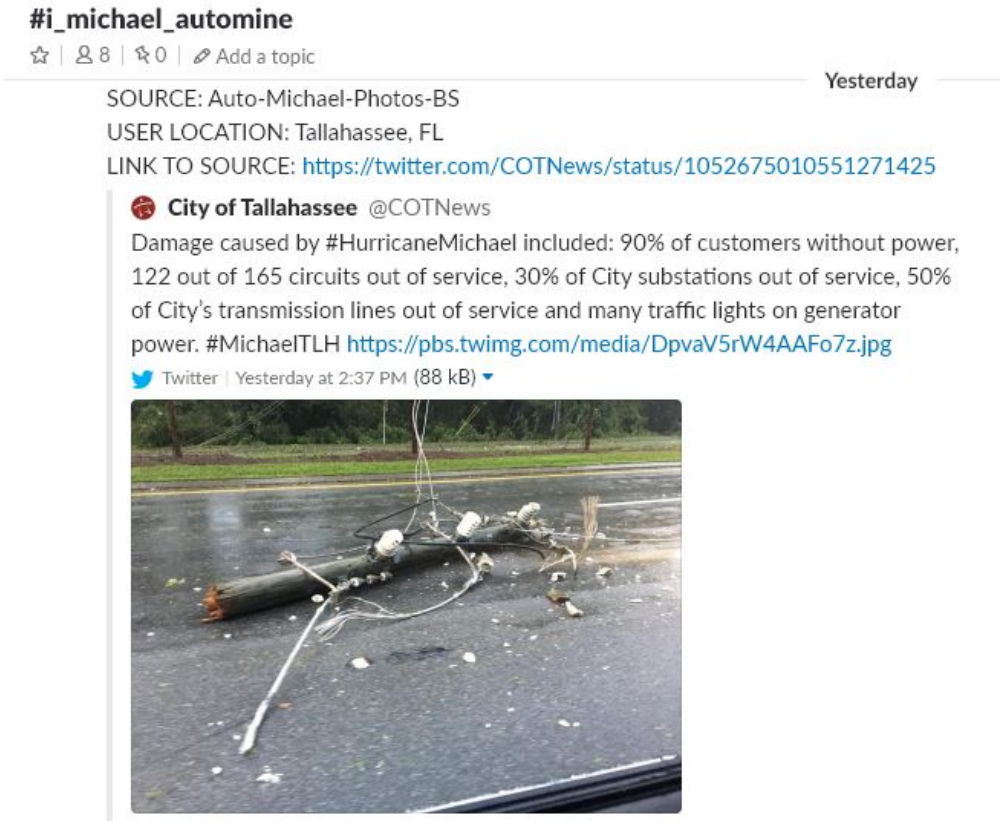
...which yields our full Twitter queries like this one:

("flooding" OR "flooded" OR "florence" OR "damage" OR "recovery") AND (Callaway OR "Cedar Grove" OR "Laguna Beach" OR "Lower Grand Lagoon" OR "Lynn Haven" OR "Mexico Beach" OR "Panama City" OR "Panama City Beach" OR Parker OR "Pretty Bayou") AND -filter:nativeretweets AND filter:images

The queries are run on the Needs_Tweets tab. Blockspring only ever returns the 100 most recent results, so we know we need 100 rows for each query. The sheet is set up therefore so search PN1 puts its results in rows 1-100, PN2 in rows 110-210, PN3 in rows 330-430, and so on. Currently the sheet is set up to execute 21 separate queries, all of which are automatically run once an hour.

So the Needs_Tweets tab always reflects the most recent search results. The Sheets function move_tweets triggers once every 30 minutes, and checks for new tweets by:
- Moving any new tweets (defined as those with a greater twitter ID than the highest last seen by move_tweets) to the Needs_Master tab
- De-duplicating the Needs_Master tab so any results from the current query which were already found will not be processed again
- Moving the actually new tweets to the Needs_Gold tab and marking the flag column on Needs_Master to indicate the tweet has been moved to gold

The final step is where Zapier is still used: a Zapier 'zap' routine is configured to monitor the Needs_Gold tab; whenever a new row appears, it executes and posts the row to CEDR's Slack automine channel. Slack conveniently automatically expands a preview of a tweet anytime a tweet url is posted to a channel, so it looks something like this:



Other Examples

The latest iteration of the automining sheet uses the same process, but searches only tweets from a Twitter list of official federal, state, and local accounts:

https://docs.google.com/spreadsheets/d/170Z2-TzQ0n9z12Ox7nluSPO7kzDyA7e1q1BB1VK1Myo/edit?usp=sharing

https://twitter.com/8bitmeme/lists/michael-official-em

The nine searches currently defined are based on FEMA's lifelines, and so when matching tweets are found, they are posted to CEDR's Slack with the additional information of which lifeline they pertain to: