December 22, 2019
LING 131: Intro to NLP with Python
Final Project
Miriam Gölz, Katie Krajovic, Angus L'Herrou, Dominique O'Donnell, and James Shatalov-Stein

A Measure for "Local-ness" of Talk Radio Conversations

What We Did and Why

Our final project for the course was an examination of a corpus to extract place names and calculate a measure of local-ness of the conversation taking place in the texts. The project was inspired by finding a rich source of material, the RadioTalk corpus. The corpus contains 6 months of automatically transcribed talk radio from 163. The text is not a continuous transcription but was recorded in 10-minute chunks.

Our group chose to consider the question of the "global conversation". We took an interest in attempting to measure how local or global the conversation taking place on the radio was—because many people listen to talk radio, and it can start conversations and raise awareness of issues. Our measure is based around mentions of geographical locations. We made this decision for several reasons: 1) It's doable, 2) it's a clear, unambiguous way to say without doubt that the conversation is about a specific place.

How We Implemented It

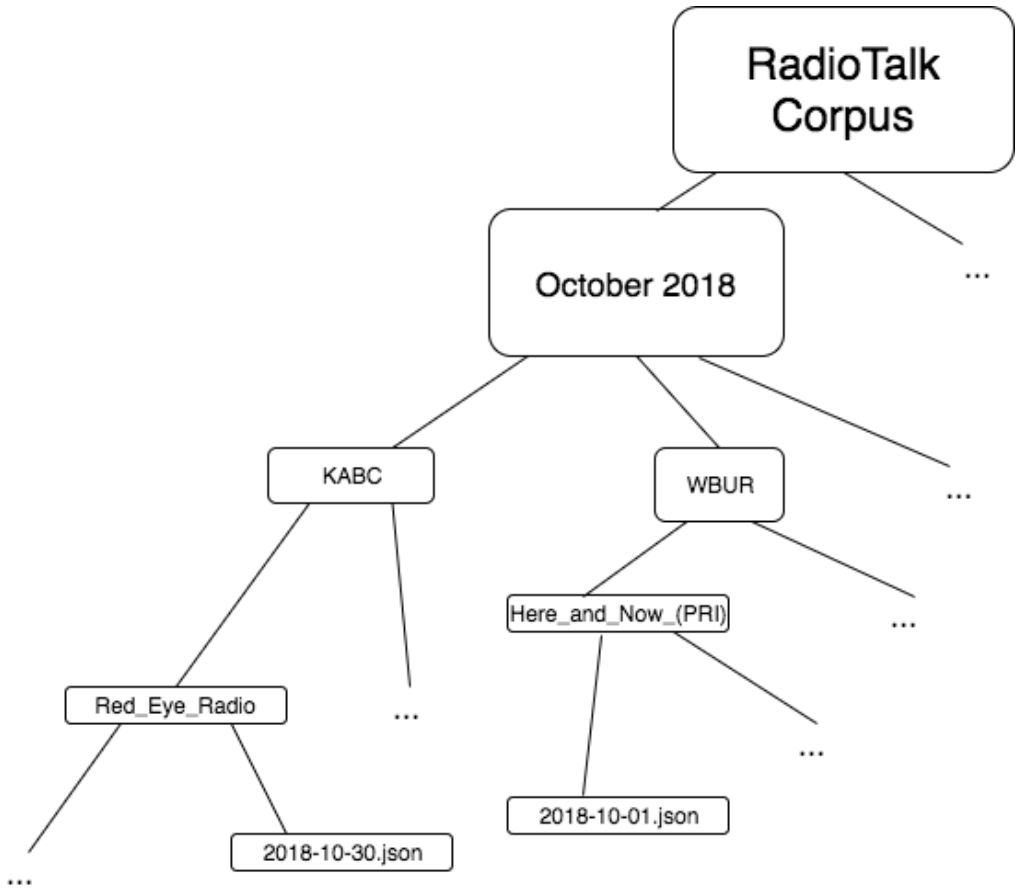
The first step in our process was to figure out how best to split the data into manageable chunks. The original corpus is available broken into months -- so we chose to start with the first month of data, October 2018. This ended up being such a huge amount of data, that we chose to restrict our study to only that data.

The corpus provides metadata about each audio chunk recorded, including the callsign of the station it was recorded from, the location of that station, and the name of the show the broadcast was part of. We leveraged that metadata to separate the data into files representing all the data from a single station, without losing any of the metadata. Then, upon further consideration, we determined that this data needed further reduction in size, so we divided the station files into files containing all audio chunks from a given show, and then each show was divided into files containing all audio chunks from that show in a given day.

However, to simplify later extraction, some metadata that did not seem relevant to us was removed when splitting the files into jsons of a single day's-worth of a show. We combined all the chunks into a single dictionary, with location, callsign, and show data, but without other information from the corpus like speaker ID, presumed gender of the speaker, start and end

time, etc. The resulting structure can be seen in the flowchart below, where our data is only from October 2018, and the internal structure of the data for the month was generated by us.

Figure 1. Corpus File Structure



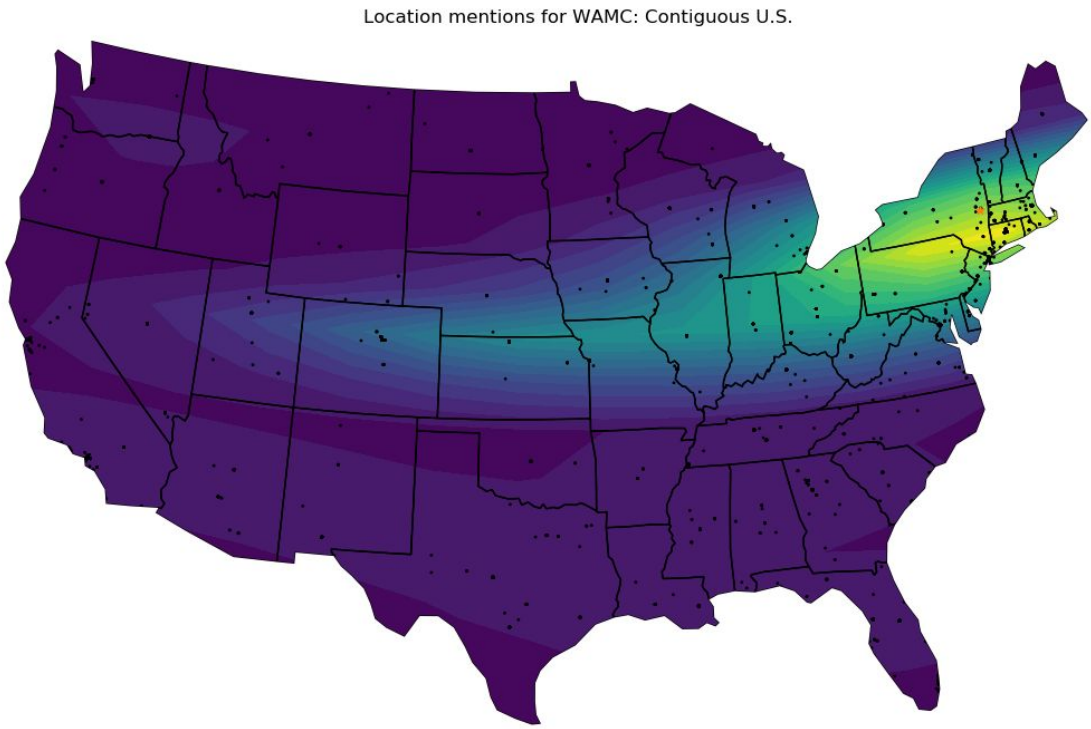
To extract locations, we began by passing a radio show's transcription to spaCy's named entity recognition model. This tokenized the string and tagged various types of entities, and we filtered out those tagged as a geopolitical entity, or "GPE" (i.e. cities, states, countries). One thing we noticed was that strings like "Boston Massachusetts" were often separated into "Boston" and "Massachusetts", and separately tagged. To remedy this, we ran a function that, for every location in a list, used the geocoding method described below to check if a) it was a state, and b) the preceding location was a city within that state. If both of these were true, we concatenated the two locations into one token.

We used the Geonames geocoder to find the geographical information about the GPEs. In order to limit the calls we made to the geocoder, and save the data relevant to us, we created a dictionary that stores each location, its country, state, and latitude and longitude. If any of this information was not applicable or available, it was simply left blank in the dictionary.

For every location tagged as a “GPE”, if it did not yet exist in our dictionary, we created an entry; but if it did, we simply extracted the data from our dictionary instead of searching it through the geocoder. Using these tools, we created a list of places mentioned in the data, each with their latitude and longitude. We additionally created a pandas data frame in order to map the data, as well as separating locations according to whether or not they were in the same and in the same country as the station broadcasting.

To represent the data we had collected for each station and each show, we generate a map of the contiguous US with a point plot and a heatmap showing the areas most talked about on that show. Currently, the plots only show those mentions that are within the contiguous US, but this could be expanded to global mentions in the future. In this case, we would like to have a better system for displaying the map; one option for this would be a web frontend. The maps are generated using `geopandas` and `geoplot`, which display geographic plots using `matplotlib`’s `pyplot` system.

Figure 2. Heatmap for WAMC Location Mentions in US Oct. 2018



As illustrated in Figure 2, the maps represent the station location with an orange star marker, and each location mentioned with a black dot.

Corpus Data

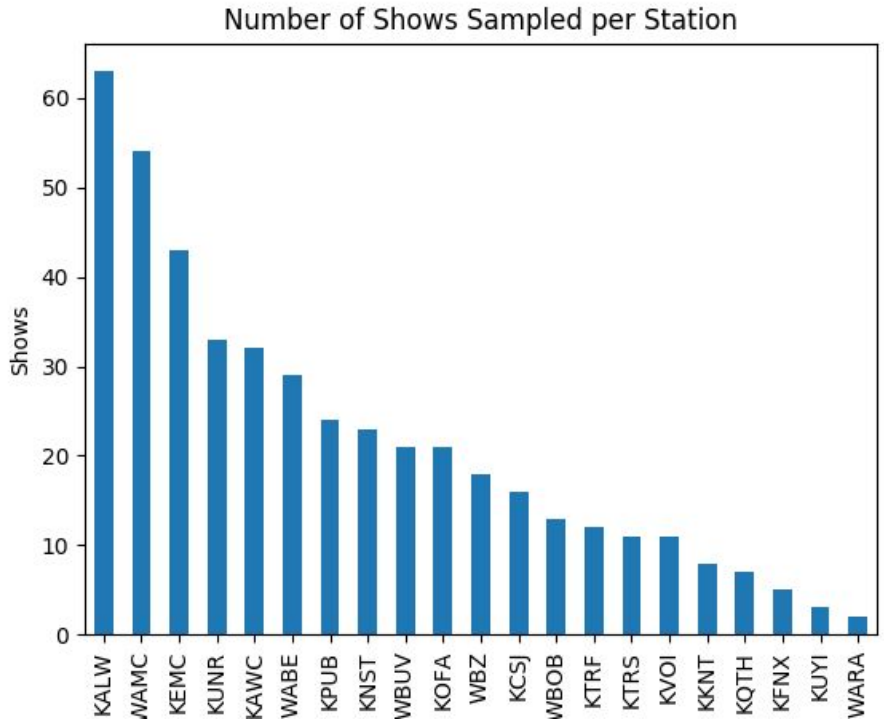
The October 2018 subset of the RadioTalk corpus includes 82 stations, but due to time restrictions we chose to include a subset and limit the number of episodes included. Our final corpus is comprised of 492 shows aired across 21 stations in October 2018. We chose to select the first 5 episodes of each show for analysis for a total of 1952 episodes. Table 1 shows the stations, station location and number of shows included in our corpus. The distributions of the data is represented in Figures 2 and 3 below.

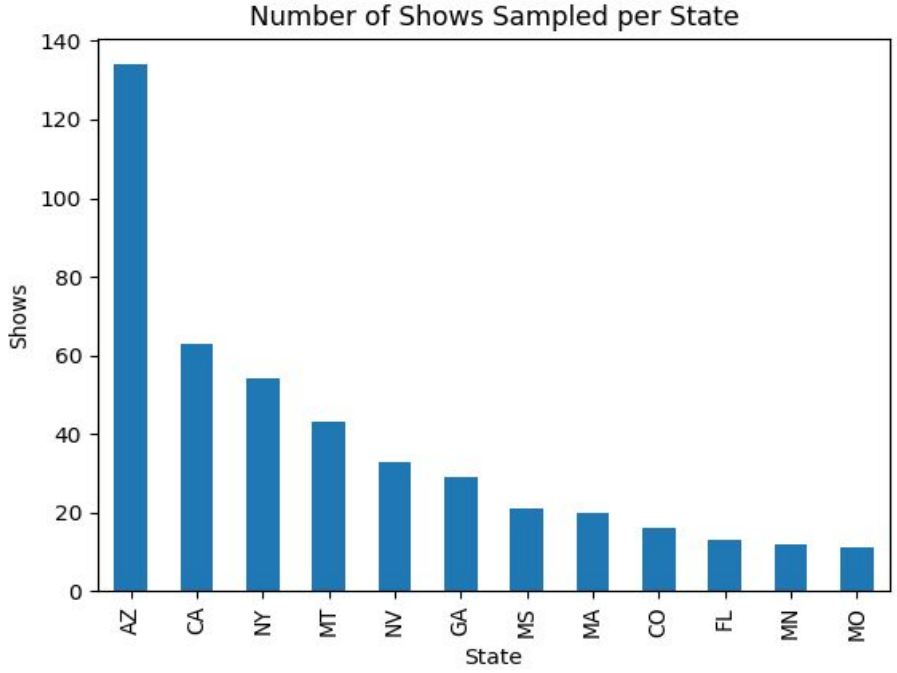
In order to identify the locations mentioned in each episode, we ran a script that analyzed the texts and concatenated new location metadata to an episode's JSON. Because the Geocoder API limits the number of requests made per hour (1000/hr), we tried adding a call to Python's `time.sleep()` as we traversed our file structure, but found that the place name dictionary (`geonamesobjs.pkl`) that we used to track identified GPS coordinates reduced the number of requests to Geonames per hour, and allowed our script to run overnight. This code is found in `fnd_all.py`.

Table 1. Shows Per Station

Station	Shows	Location	Station	Shows	Location
KCSJ	16	Pueblo, CO	KUYI	3	Kearn Canyon, AZ
WBOB	13	Jacksonville, FL	KFNX	5	CaveCreek, AZ
KNST	23	Tucson, AZ	WBZ	18	Boston, MA
KEMC	43	Billings, MT	KTRS	11	St Louis, MO
WAMC	54	Albany, NY	KVOI	11	Yuma, AZ
WARA	2	Attleboro, MA	KUNR	33	Reno, NV
KALW	63	San Francisco, CA	KAWC	32	Yuma, AZ
KQTH	7	Tucson, AZ	KOFA	21	Yuma, AZ
KPUB	24	Flagstaff, AZ	KTRF	12	Thief River Falls, MN
WABE	29	Atlanta, GA	WBUV	21	Biloxi, MS
KKNT	8	Phoenix, AZ			

Figures 3 & 4. Distribution of Shows





For each radio station, we collected counts of the number of tokens, the number of tokens that were tagged as locations, how many of these locations were in-state and out-of-state (relative to the location of the radio station), and how many of these locations were inside and outside the country. We used `total_counts.py` to retrieve these counts and compile them into `all_station_counts.csv`. The first line of this file is shown below: for each category (in state, out of state, in the country, out of the country), we calculated both the percent of locations as well as the percent of tokens that was made up by that category.

Table 2. Sample Station Row Data for KALW

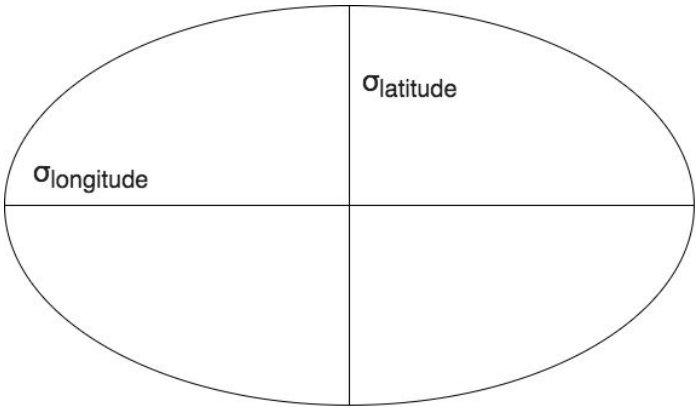
	Station Location	Token Count	Location Count	Location %	In State	In State % Loc	In State % Tok
KALW	San Francisco, CA	1426385	12090	0.85	1233	10.2	0.09

Out of State	Out of State % Loc	Out of State % Tok	In US	In US % Loc	In US % Tok	Out of US	Out of US % Loc	Out of US % Tok	Dispersion
10857	89.8	0.76	5987	49.52	0.42	6103	50.48	0.43	4499

Locality

As a measure of locality, we've adapted a metric from crime statistics. It's rather rudimentary, but it is a good initial measure. It is the area of the ellipse formed by using the standard deviations of the latitudes and longitudes of the locations mentioned in shows as axes.

Figure 5. Visualization of Locality Measure



What this is specifically measuring then, is how close or far the locations are to their mean location — as locations skew further and further away from the mean, the measure grows, and so a high dispersion, as with KALW below, indicates a wider scope of conversation, geographically speaking, and a lower dispersion, as with KCSJ, indicates a more local conversation, where all the places mentioned are clustered close together.

KCSJ, Pueblo, CO: 2863, locations per 1000 tokens: 5
WBUV, Moss Point, MS: 2935, locations per 1000 tokens: 8
KKNT, Phoenix, AZ: 3101, locations per 1000 tokens: 7
WBZ, Boston, MA: 3168, locations per 1000 tokens: 8
KFNX, Cave Creek, AZ: 3193, locations per 1000 tokens: 7
KTRS, Saint Louis, MO: 3212, locations per 1000 tokens: 9
WBOB, Jacksonville, FL: 3302, locations per 1000 tokens: 8
KNST, Tucson, AZ: 3347, locations per 1000 tokens: 6
KQTH, Tucson, AZ: 3456, locations per 1000 tokens: 7
KVOI, Cortaro, AZ: 3487, locations per 1000 tokens: 8
KOFA, Yuma, AZ: 3492, locations per 1000 tokens: 10
WABE, Atlanta, GA: 3541, locations per 1000 tokens: 10
KTRF, Thief River Falls, MN: 3575, locations per 1000 tokens: 7
WAMC, Albany, NY: 3669, locations per 1000 tokens: 10
KEMC, Billings, MT: 3768, locations per 1000 tokens: 10
KUNR, Reno, NV: 3951, locations per 1000 tokens: 11
KAWC, Yuma, AZ: 3967, locations per 1000 tokens: 12
WARA, Attleboro, MA: 4019, locations per 1000 tokens: 5
KPUB, Flagstaff, AZ: 4089, locations per 1000 tokens: 12
KALW, San Francisco, CA: 4499, locations per 1000 tokens: 8
KUYI, Hotevilla, AZ: 4840, locations per 1000 tokens: 5

These dispersions are listed with the number of locations per 1000 tokens, because a mere wide net of locations is not sufficient to capture the “globality” or “locality” of a conversation—without considering this statistic, one could have a conversation in which the only locations referenced are the magnetic poles, once each, and that conversation would have a dispersion of 45,580. For this reason, it’s important to view these data in the context of other, similar texts, and with the other metrics we examined.

Running the Main Script

To run our main script, set up a clean Conda Python 3.7 interpreter and run the following:

```
$ conda config --add channels conda-forge
$ conda config --set channel_priority strict
```

```
$ conda install -y -c conda-forge geoplot
$ python main.py
```

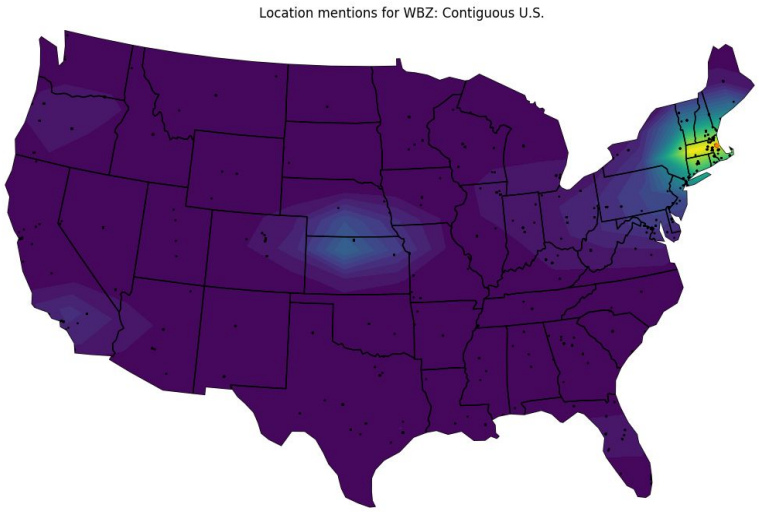
When the main script is run, the user is asked to choose a radio station, and presented with some counts and percentages, as shown below for WBZ:

Figure 6. Count Statistics Output for WBZ

	Station Location	Total Count	In State	In State %	Out of State	Out of State %	In US	In US %	Out of US	Out of US %
WBZ	Boston, MA	7235	1718	23.75	5517	76.25	4925	68.07	2310	31.93

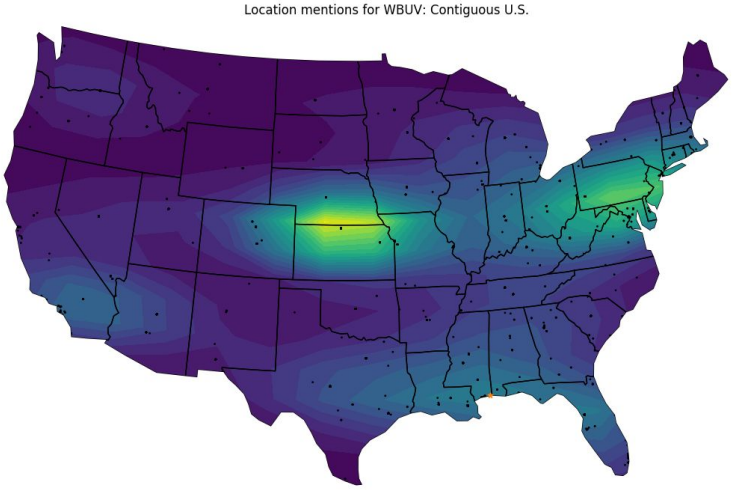
Additionally, a heatmap is generated and saved in the “maps” directory. As seen in the map for WBZ below, being a Boston-based station, many locations mentioned are very local to the New England area. There is also a lighter spot in the middle of the country, which is most likely due to mentions of the United States, as the latitude and longitude defined for the country were its geographic center.

Figure 7. Heatmap for WBZ Boston Location Mentions in US Oct. 2018



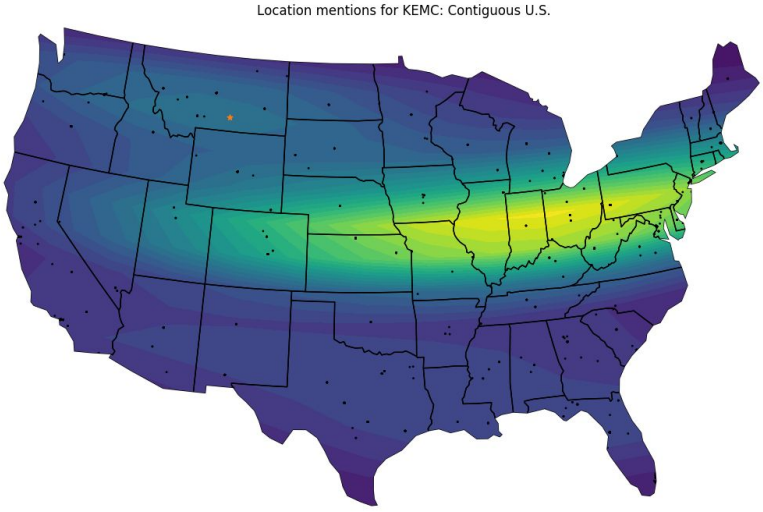
Not all radio stations show such a local distribution of location mentions. For example, WBUV in Biloxi, MS seems to mention the United States as a whole more often than local locations:

Figure 8. Heatmap for WBUV Location Mentions in US Oct. 2018



And KEMC located in Billings, MT, had location mentions more widespread across the country:

Figure 9. Heatmap for KEMC Location Mentions in US Oct. 2018



Hiccups and Hoorays

Our group ran into some hiccups along the way to our final results. One such issue was the frequent FCC mandated recitation of callsigns and station locations. This skewed our data toward local-ness where the topics of conversation may otherwise be more broadly dispersed. We considered masking locations within a few tokens of the callsign, but this became too

cumbersome, given our time constraints. This, therefore, is an unresolved issue, that could be easily resolved, given a little more time. It would involve additional preprocessing—tokenizing, removing tokens around call signs, and re-concatenating into a string each ‘episode’s’ content, and given the sheer amount of our data, this is unrealistic to do at scale.

Another problem that was a stumbling block for us was choosing the correct way to look up longitudes, latitudes, and other location data. Initially, we were using a service called Nominatim, which Miriam has found, but the service ultimately didn’t work for us, because it could not handle the frequency and amount of our requests; furthermore, we later learned that they specifically ask users not to use their service on large-scale projects. Thus, the search began for a new one. Fortunately, at Marc’s suggestion, Katie researched access to geonames, a similar service with limits on the number of requests that can be made per hour but which does not seem to mind being used for a large project. In order to do our best to nullify that restriction, we devised a plan to create a project-internal reference dictionary that would contain all previously looked-up locations with the relevant data associated with them (i.e. longitude and latitude, state, country, etc.). This would then be the primary source for looking up locations—rather than the online service.

If this project were to be continued, we would want to explore the possibility of self-hosting an instance of Nominatim, Geonames, or another similar service, as the API limits restricted our ability to process the corpus data. Having control over the source would allow for more flexibility in which service to choose as well as how carelessly we could query it.

Who did what?

Our work together was highly cooperative, sometimes with two people coding together, so some of the work was done by subgroups of our group. Katie was the one to find the RadioTalk Corpus. Dom played a large role in fine-tuning our research questions and methods to try better to capture exactly what we are interested in. She and James worked together to design the structure for the data. James wrote and ran the code to actually separate out stations and then shows and then show episodes from the corpus data. Miriam and James worked together on methods for fetching and calculating statistics for stations and shows. Miriam and Katie worked together on finding and learning to use the named entity extraction model and also on correcting the “city, state” issue. Miriam and Katie also cooperated on finding and learning to use the web services that allowed us to check longitudes and latitudes. Angus was an all-around helper, and also is the group’s cartographer. He learned to generate our maps, and was the go-to-guy for GitHub issues, among other things. Angus and Miriam also worked together on the main script. Dom took the location extraction method and modified it to run at scale.

Scripts

The interactive portion of our code consists of the following scripts:

- `main.py`
  - Browser of the `data` folder; calls the following three scripts to generate information about a station or show.
- `radiomap.py`
  - Given a dataframe of latitudes and longitudes of place mentions, generates a PNG heatmap of the points that fall in the contiguous United States.
- `count_locations.py`
  - Module that contains methods for calculating some statistics about the location mentions, such as how many are in-state versus out-of-state or how many are in the US versus international.
- `create_dataframe.py`
  - Module that contains methods for making dataframes for stations and shows from the JSON files for each show. These dataframes are used for generating the heatmaps with `radiomap.py`.

Other scripts we wrote that were used to interact with the corpus, prepare the data, and interface with Geonames are as follows:

- `find_all.py`
  - Script traverses corpus subfolders and calls methods to retrieve locations for the first five episodes of each show. Script will traverse entire corpus.
- `finalfindlocs.py`
  - Given a document, script identifies location mentions in a document text. Retrieves latitude and longitude of location names from Geocoder. Updates a dictionary of found locations and their GPS coordinates. Adds location information to document JSON and concatenates all episode JSONs for a given show.
- `makeepisodefile.py`
  - Was used to take json files with multiple audio chunks and all their meta-data, all from the same show on the same station, and the script created folders of the show names and wrote files of all the text uttered in the show on a given day.
- `total_counts.py`
  - Script generates a .csv of token and location counts by radio station, counts and gives the percentages for locations inside and outside the station's city and country, as well as dispersion.
- `file_checker.py`
  - Script checks station subfolders of the corpus for any misplaced files and prints their path.
- `file_cleanup.py`
  - Deletes files in station subfolders which do not belong there after a previous error which caused files to be generated with incorrect names.

References

Beeferman, D., Brannon, W., & Roy, D. (2019). RadioTalk: a large-scale corpus of talk radio transcripts. *arXiv preprint arXiv:1907.07073*.

Gölz, M., K. Krajovic, A. L'Herrou, D. O'Donnell, and J. Shatalov-Stein (2019). *ling131-project*. https://github.com/angus-lherrou/ling131-project.

Levine, Ned (2010). *CrimeStat*: A Spatial Statistics Program for the Analysis of Crime Incident Locations (v 3.3). Ned Levine & Associates, Houston, TX, and the National Institute of Justice, Washington, DC. July.

Social-Machines. (2019, November 6). social-machines/RadioTalk. Retrieved from https://github.com/social-machines/RadioTalk.