Joshua Stapley, Net ID: js83, Weather Data Progress Report

Introduction

As I've explored the National Weather Service (NWS) API and discovered what can be accomplished with it, I've shifted my focus from one application of text retrieval and analysis to another. The original plan to create a search engine for NWS documents wouldn't provide much value, especially because it turns out there is a robust search system already in place, and I had no ideas for improving it. Instead, I will be analyzing the sentiment toward the weather in cities across the United States.

The end goal is to create an application where a city can be searched or selected from a dropdown menu, recent weather for that city will be displayed, and sentiment analysis will be performed on Twitter data related to the weather for that city. Currently, little progress has been made, but a firm plan is in place for an initial working solution.

Completed Tasks

Reading about the NWS API led me to the National Oceanic and Atmospheric Administration (NOAA) SDK developed by Paulo Kuong (https://github.com/paulokuong/noaa). This kit, developed for Python, serves as a generic wrapper for the NWS API and is updated as the API changes. Using this SDK, I input a zip code (as the frontend is not yet developed, this input is generated from the console) and information for the nearest weather observation station is returned in JSON format. By default, hourly weather data from the last 7 days is returned. This returned data is parsed and stored in a pandas dataframe consisting of the timestamp, station acronym, and the following weather data for each hour: text description of the weather, temperature, wind speed, precipitation, relative humidity, wind chill, heat index, and cloud layers. This data will be used to determine what the weather has been over the course of the preceding week.

Twitter data will be returned using the tweepy library. By default, only tweets from the preceding 7 days are returned in response to a request, so it lines up nicely with the output from NWS. Currently, I generate a second input from the console, and tweets related to that query are returned. The query I generate consists of the name of a city followed by some weather term, such as "Houston weather," "Boston snow," or "Tulsa cold." The content of the resulting tweets is extracted and stored in a list. This will be the data on which sentiment analysis will be performed.

Pending Tasks

The frontend needs to be developed. It will be a very simple JavaScript application. Initially, a dropdown menu including a list of limited cities to be selected will be used. Once a city has been selected, a submit button can be pressed and a query generated. This query will be integrated with the existing Python code. Initially, a dictionary will be used to look up the city name and return a zip code, which can then be used to return 7-day historical weather data. If the application is expanded to include many cities, I may leverage a geocoding API to accomplish this step.

The city query will also be used to automatically generate multiple queries for Twitter data. As mentioned before, these queries will all contain the city name and some weather-related term, but multiple city/term combinations will be used to generate one request for each combination, which will yield ~10 requests for Twitter data for each query. The terms used remains to be decided. They may

come from a static list of weather terms, the same no matter the city or recent observed weather conditions. Alternatively, the terms used may come from the text descriptions of the weather from the NWS API. These text descriptions contain terms and phrases such as “Mostly Cloudy,” “Heavy Rain and Fog/Mist,” “Light Snow,” “Hazy,” etc… This may help us return more relevant tweets than the static list. The methods could be combined as well.

Once the Twitter queries have been generated and tweets returned, duplicate tweets will be deleted, and sentiment analysis will be performed using the text of the tweets. Initially, this analysis will be done using existing sentiment analysis libraries for Python, most likely Text Blob or VADER.

Once the sentiment analysis has been performed for a query, the sentiment score will be returned and displayed in the frontend, as well as a term describing this sentiment score and a word cloud, or something similar, with common words used to describe feelings about the weather.

Challenges

The main challenge for which I haven’t yet come up with a solution is how to further clean the data. It’s possible many tweets with little relevance could be returned, injecting substantial noise into the model. As I have yet to implement the sentiment analysis, I don’t know how much of an issue this will be.

I also have concerns that I will need to generate more requests for Twitter data than initially anticipated to return a dataset of decent size for each city query. If that ends up true, I may hit rate limits.

The above issues present some chance that the final product will be practically useless, but I’m currently optimistic the results won’t be that bad.

One final issue is that the NWS API currently has some bugs and limitations. The precipitation values returned by the API don’t match the observations found on the corresponding weather stations’ websites. The API only returns 0 or “None” for these values. I have contacted support to get this fixed. As for limitations, there is some weather data I would like to include that isn’t extractable using the API, such as snowfall amounts. I’ve come across another API from OpenWeatherMap that may provide this data. I may decide to use this API instead.

Conclusion

Many tasks remain but the blueprint is in place for a preliminary application. If the results are promising, the application could be expanded to include a much longer history of weather and Twitter data and could be used to gauge general sentiments among the population of a city (not only sentiments coming from tweets explicitly discussing the weather), and if and how those general sentiments are affected by weather patterns in a city.