Lab Report

Title: GIS 5572 lab1- API's
Notice: Dr. Bryan Runck
Author: Mary Heise
Date: Feb 11, 2021

Project Repository: *https://github.com/Tuletara/GIS5572/tree/main/Lab1*

Abstract

Big data is only a resource if it can be efficiently utilized. The construction of Extract-Transform-Load (ETL) pipelines in conjunction with Application Programming Interfaces make this possible. The key is knowing how to decompose the sites that house the data and the ability to write a program that can sift through the site structure in order to collect the useful data. Three websites, each with a different layout and data types are assessed here. Python Jupyter Notebook was used to write scripts to pull data based on the methodologies of CKAN and Google Places APIs. The outputs are standard data formats for spatial and tabular data: shapefiles, comma separated value files, and json. These outputs were generated from the websites using three different constructed ETLs which were uploaded to GitHub. The notebooks and files are functioning resources that can be reused and modified to harvest more data. Although the end result is a useful tool, creating them did not save time over directly interfacing with each website and windows file explorer. This is due to a steep learning curve to understand how the process works as well as deconstructing the sites used.

Problem Statement

There is a massive amount of information available to utilize and analyze geospatially. Much of this data is freely available on the web. In order to efficiently disseminate these vast data stores Extract-Transform-Load (ETL) processes should be used. This requires an extensive understanding of the layout of the website and some programming ability. Many sites offer a standardized interface known as an Application Programming Interface (API) that abstracts away most of the coding needed to scrape the data. The process of collecting data through an ETL, written in Python and Jupyter Notebooks, using the API's of Minnesota Geospatial Commons, Google Places, and North Dakota School of Agriculture (NDAWN) websites is described in this report. All three sites use very different interfaces so a comparison of the site lay-outs will also be discussed.

Input Data

The data being explored varies depending on the source and the content is arbitrary given the goal of this lab as a means to collect data. In general the ETL will grab spatial data in the form of a zipped shapefile from the MN Geospatial website. A location search will be queried Google places and the results returned in Jupyter Notebook in a json format. Temporal and station specific data tables will be saved from the NDAWN site as a .csv document.

Table 1. Overview of websites and data types used.

#	Source & Link	Spatial Data
1	Minnesota Geospatial Commons	Various geospatial data and formats will be searched & downloaded in a zipped shapefile.
2	Google Places	Location search will yield results in json format.
3	NDAWN	Weather data saved as a .csv.
4	Jupyter Notebook	Repo posted to GitHub

Methods

Each website must be analyzed to determine how the data is structured. Site specific API documentation is accessible for both MN Geo and Google, while NDAWN analysis is dependent on breaking down the url and inspecting the source code. The best way to decompose the layout is to open up links and tables for different weather criteria to find the pattern in the url. Google Places has a clear and straightforward guide on how data requests need to be formatted. MN Geo has a link to the CKAN API that site uses. CKAN has demo action lists that can be modified to specific search criteria.

The following conceptual models were distilled using the preceding methods and coded using Python sherpa, requests, pandas, json, and urllib libraries. For MN Geo, the first step is to get a list of package tags which can be used to search for the subject of data; water, roads, boundaries, etc. A quality check for server response is confirmed and a list of tags is returned. From here, a url is generated and the file of interest is download locally with requests.get(). With Google Maps API, the first step is to create an account, complete with your bank information, to acquire an API key. After this step a url can be generated by specifying the required parameters, location information, and urlencode. The url is sent over to the server and the results returned with r.json(). To obtain NDAWN data, a list of url's was first generated for each weather station then date and type criteria were added on. Requests.get was used to ping the server and open().write delivered the files locally.

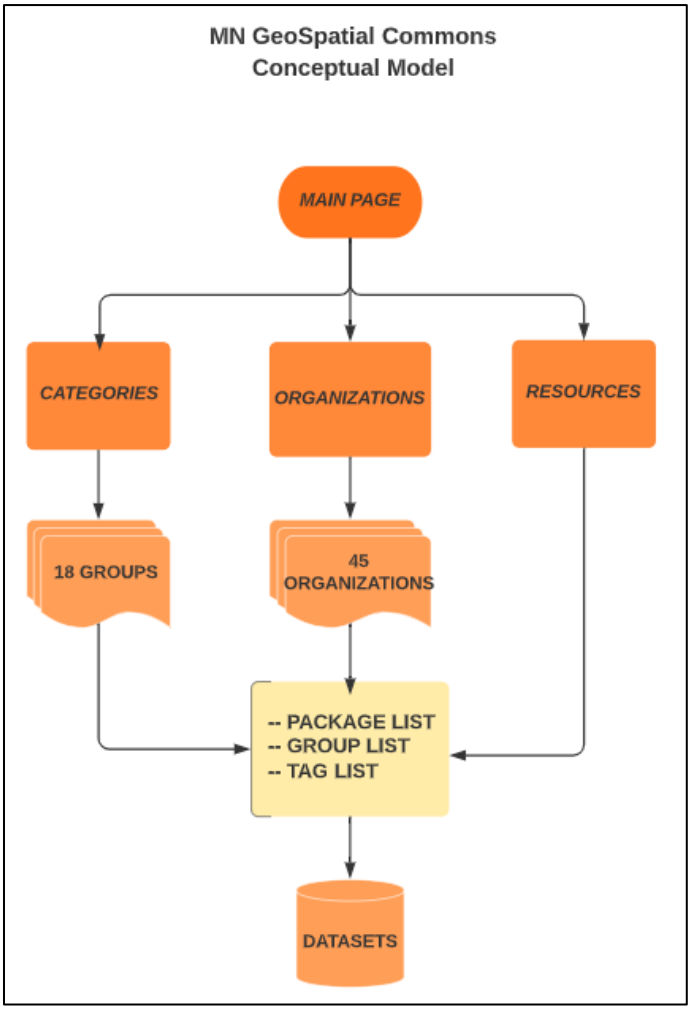

Figure 1. MN Geospatial Commons API model.

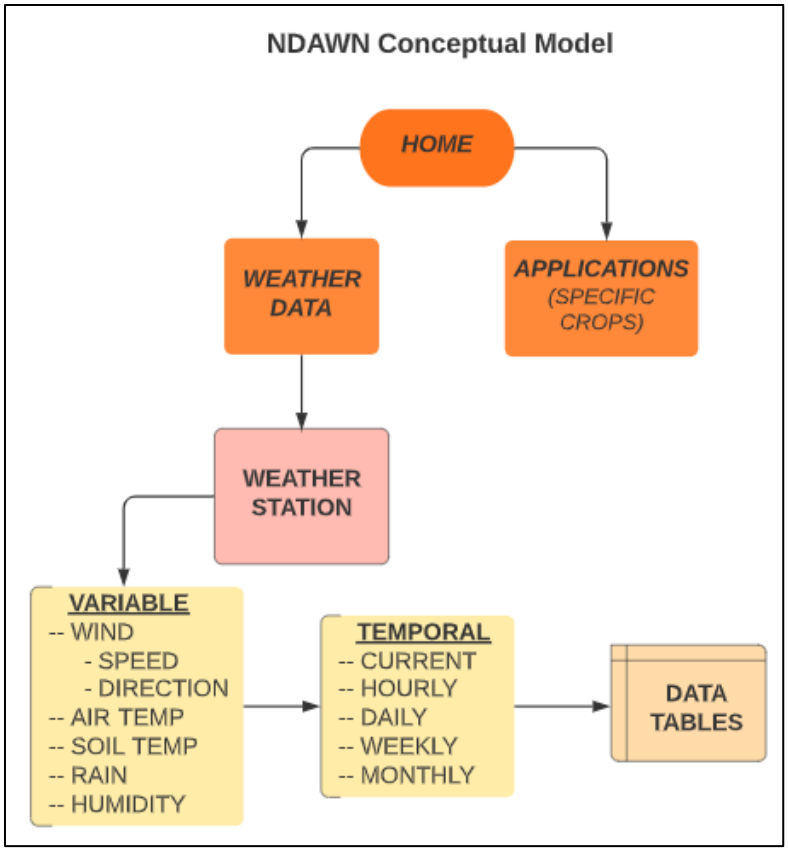

Figure 2. NDAWN web layout.

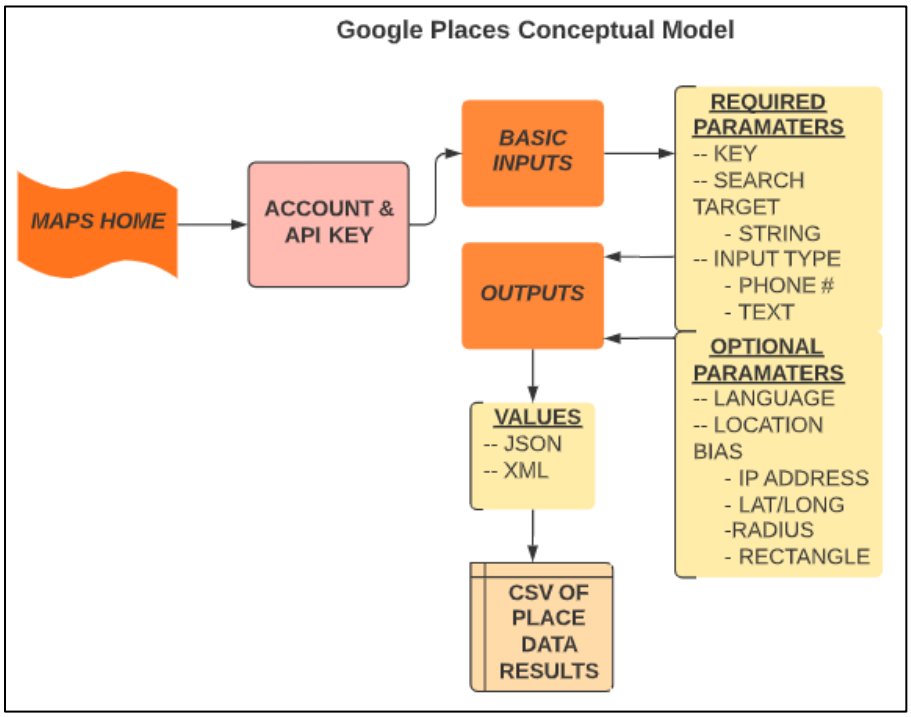

Figure 3. Google Places API model.

Results

In each instance, data was successfully delivered in the desired format. The standard zipped .shp, .csv, and json were pulled through the ETL, stored locally where needed, and can now be used for analysis (or to find a tasty restaurant at lunch time!). Additionally, the code was run a second time through and new data was generated. All three notebooks have been placed in my GitHub repo.

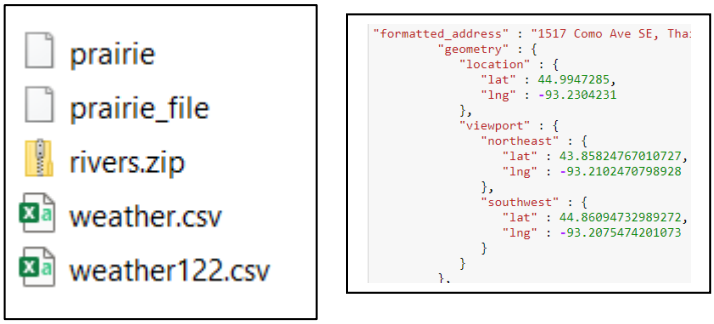

Figure 3. ETL outputs: data files downloaded to local drive & json

Results Verification

The downloaded files from MN Geo and NDAWN are workable files. The zipped file can be extracted and imported into a GIS and the weather data file can be opened and interrogated. Finally, the Google API returned restaurant suggestions in a json format. The ETL has been confirmed by replicating the process with new datasets.

Discussion and Conclusion

There are a multitude of options for accessing the wealth of freely available data online and each method has unique features. API's can streamline the ETL process but there will still be a lot of variability in how the data is searched, gathered, and saved. I did not find the CKAN API particularly user friendly but this is likely due to my own inexperience. Most developer documentation is not written for beginners and the plethora of modules available is a guarantee that some method will succeed but a fair amount of troubleshooting will be required. The difficulty for me was not knowing how to evaluate the errors, gaps in code, or even what to search for to resolve the issue. That being said, the CKAN site did lay out demo code plainly.

Google's API was largely a step by step process. Handing over your credit card to get an API key was not a comfortable situation. There are concerns about a massive tech company having more personal and financial information and, not knowing anything about this process, what exactly can turn out to be billable.

Then NDAWN site turned out to be pretty simple to distill once I understood how to look for patterns. Again, the difficulty came from not knowing how to code it. Luckily, getting text files doesn't take a lot of code. I did pour through several online resources in order to learn about ETLs and APIs. Many were helpful but it's definitely a piecemeal process that requires perseverance, patience and a basic knowledge of programming. In a big data world however, these efforts will pay dividends by allowing for an efficient and mostly hands-off approach to data collection.

References

CodingEntrepreneurs. “30 Days of Python - Day 20 - Using Google Maps Geocoding and Places API - Python TUTORIAL.” *YouTube*, 1 May 2020, www.youtube.com/watch?v=ckPEY2KppHc&list=PLAJR__ussXqVPVMsuUcDdRI-MzLikySwV&index=11&t=464s. Accessed 12 Feb. 2021.

curl. “Curl - Tutorial.” *Curl.se*, 2021, curl.se/docs/manual.html. Accessed 7 Feb. 2021.

Data Warehouse Guide. “3 Ways to Build an ETL Process.” *Panoply*, 2018, panoply.io/data-warehouse-guide/3-ways-to-build-an-etl-process/.

linuxize. “How to Make a POST Request with CURL.” *Linuxize.com*, 23 July 2020, linuxize.com/post/curl-post-request/. Accessed 7 Feb. 2021.

---. “Using Curl to Make REST API Requests.” *Linuxize.com*, 27 May 2020, linuxize.com/post/curl-rest-api/. Accessed 7 Feb. 2021.

Python. “Ftplib — FTP Protocol Client — Python 3.9.1 Documentation.” *Docs.python.org*, 2021, docs.python.org/3/library/ftplib.html#ftplib.FTP.nlst. Accessed 8 Feb. 2021.

---. “Urllib.parse — Parse URLs into Components — Python 3.8.3 Documentation.” *Docs.python.org*, 2021, docs.python.org/3/library/urllib.parse.html.

python programming. “Python Programming Tutorials.” *Pythonprogramming.net*, 2021, pythonprogramming.net/ftp-transfers-python-ftplib/. Accessed 8 Feb. 2021.

PYthon Tutorials. “Requests: HTTP for Humans.” *Pythonspot*, 2021, pythonspot.com/requests-http-for-humans/. Accessed 10 Feb. 2021.

Python, Real. “Python’s Requests Library (Guide) – Real Python.” *Realpython.com*, 2021, realpython.com/python-requests/.

PythonSherpa. “PythonSherpa.” *Www.pythonsherpa.com*, 2021, www.pythonsherpa.com/tutorials/2/. Accessed 11 Feb. 2021.

Schafer, Corey. “Python Requests Tutorial: Request Web Pages, Download Images, POST Data, Read JSON, and More.” *YouTube*, 26 Feb. 2019, www.youtube.com/watch?v=tb8gHvYlCFs.

Simplified Python. “Python Download File Tutorial - How to Download File from Internet Using Python.” *Simplified Python*, 2021, www.simplifiedpython.net/python-download-file/.

stackoverflow. “Wget - Download All Files in a Path on Jupyter Notebook Server.” *Stack Overflow*, 2021, stackoverflow.com/questions/43042793/download-all-files-in-a-path-on-jupyter-notebook-server. Accessed 8 Feb. 2021.

Traversy Media. “What Is a RESTful API? Explanation of REST & HTTP.” *YouTube*, 20 Feb. 2017, www.youtube.com/watch?v=Q-BpqyOT3a8. Accessed 8 Feb. 2021.

Tutorialspoint. “Downloading Files from Web Using Python?” *Www.tutorialspoint.com*, 2021, www.tutorialspoint.com/downloading-files-from-web-using-python. Accessed 7 Feb. 2021.

Vaderia, Shyamal. “Downloading and Unzippig a Zip File.” *Filling the Gaps*, 18 Aug. 2017, svaderia.github.io/articles/downloading-and-unzipping-a-zipfile/. Accessed 11 Feb. 2021.

Self-score

Fill out this rubric for yourself and include it in your lab report. The same rubric will be used to generate a grade in proportion to the points assigned in the syllabus to the assignment.

Category	Description	Points Possible	Score
Structural Elements	All elements of a lab report are included **(2 points each)**: Title, Notice: Dr. Bryan Runck, Author, Project Repository, Date, Abstract, Problem Statement, Input Data w/ tables, Methods w/ Data, Flow Diagrams, Results, Results Verification, Discussion and Conclusion, References in common format, Self-score	28	28
Clarity of Content	Each element above is executed at a professional level so that someone can understand the goal, data, methods, results, and their validity and implications in a 5 minute reading at a cursory-level, and in a 30 minute meeting at a deep level **(12 points)**. There is a clear connection from data to results to discussion and conclusion **(12 points)**.	24	20
Reproducibility	Results are completely reproducible by someone with basic GIS training. There is no ambiguity in data flow or rationale for data operations. Every step is documented and justified.	28	25
Verification	Results are correct in that they have been verified in comparison to some standard. The standard is clearly stated **(10 points)**, the method of comparison is clearly stated **(5 points)**, and the result of verification is clearly stated **(5 points)**.	20	16
		100	89