HARVESTING, INTEGRATING AND DISTRIBUTING LARGE OPEN GEOSPATIAL DATASETS USING FREE AND OPEN-SOURCE SOFTWARE

Ricardo Oliveira[a]
Rafael Moreno[b]
University of Colorado Denver, Department of Geography and Environmental Sciences, USA
[a] Ricardo.Oliveira@ucdenver.edu
[b] Rafael.Moreno@ucdenver.edu

Commission VII, SpS10 - FOSS4G: FOSS4G Session (coorganized with OSGeo)

KEY WORDS: Open Data; Cloud services; geospatial Open-Source Software; large data sets.

ABSTRACT:

Federal, State and Local government agencies in the USA are investing heavily on the dissemination of Open Data sets produced by each of them. The main driver behind this thrust is to increase agencies' transparency and accountability, as well as to improve citizens' awareness. However, not all Open Data sets are easy to access and integrate with other Open Data sets available even from the same agency. The City and County of Denver Open Data Portal distributes several types of geospatial datasets, one of them is the city parcels information containing 224,256 records. Although this data layer contains many pieces of information it is incomplete for some custom purposes. Open-Source Software were used to first collect data from diverse City of Denver Open Data sets, then upload them to a repository in the Cloud where they were processed using a PostgreSQL installation on the Cloud and Python scripts. Our method was able to extract non-spatial information from a 'not-ready-to-download' source that could then be combined with the initial data set to enhance its potential use.

1. INTRODUCTION

The availability and accessibility to Open-Data sets has exploded in recent years caused primarily by the ease that organizations now have in exposing such datasets to citizens, and also by the increasingly interest of the general public to explore these datasets. However, some barrier still exists, such as not all datasets are readily available for some custom applications that require integration of information from multiple data sets. It was with this challenge in mind that we explored ways to integrate information from different open data sets to facilitate new uses of these data.

For our tests we chose two data sets. First, the parcel data set provided freely by the City and County of Denver (http://data.denvergov.org/dataset/city-and-county-of-denver-parcels). This dataset contains 224,256 records corresponding to each parcel in the city and it is part of a larger collection of open data offered by the city through their Open Data Catalogue (http://data.denvergov.org/). It contains numerous attributes of interest and it can potentially be used for a broad range of applications. Second, we selected the tax property records provided by the Denver Property Taxation and Assessment System. This system provides a wide range of information for each parcel in the city (224,256 of them). We identified that some pieces of information available through the tax property system are not present in the parcels data set available through the city's Open Data Catalog.

2. GOAL

Our goal was to demonstrate the use of Free and Open-Source Software (FOSS) to harvest, integrate, and distribute a custom version of the parcel data set with added information extracted from the Denver Property Taxation and Assessment System. Although there are multiple sophisticated ways to achieve this goal, we concentrated in using mature well-known FOSS and procedures that are as simple as possible with the aim of demonstrating that users with a minimal set of skills can develop similar applications to fulfil their specific needs.

3. METHODS

We used two FOSS: PostgreSQL-PostGIS and Python. Both of these FOSS are mature, well-document and have multiple free resources to learn them. PostgreSQL-PostGIS was used for data storage and management providing a solid and flexible base to support any distribution system that we may decide to use to share our data. For data processing we decided to use Python given its flexibility and power.

The first step was to test that through simple procedures that both data sets (parcel data and tax records) can be integrated. To do this, we created a free version of an Amazon Web Services Relational Database Service (AWS-RDS) (https://aws.amazon.com/rds/) that uses PostgreSQL as its engine. This PostgreSQL instance was enabled with the PostGIS spatial extension (http://postgis.net/). The city's parcel data in ESRI shapefile was downloaded to a local computer, and the PostGIS shp2pgsql loader utility (http://postgis.net/docs/manual-1.3/ch04.html#id435700) was used to convert the shapefile to a spatial table in the PostgreSQL-PostGIS instance running in the AWS-RDS. The city parcel's unique ID (known as the PIN number) exists in both the tax records and the parcels table and we used it to test that the records from both tables are correctly matched. A Python script was written to: a) connect to the PostgreSQL-PostGIS instance running in AWS-RDS; b) create and execute a SQL statement that retrieves all the parcel's unique ID (known as the PIN number) from the parcels table just created in the previous step; c) structure an URL for the specific entry at the Denver Property Taxation and Assessment System

corresponding to each parcel; d) then the script parses the page's HTML code to extract any attribute of interest for each parcel registered in the tax system (e.g. "chain of ownership" that does not exist in the parcel data set), this approach is referred to as "Web Scrapping" (for definition see https://en.wikipedia.org/wiki/Web_scraping); e) finally, the information scrapped from the tax system is inserted into a record in a new PostgreSQL table created in the AWS-RDS. At this point we have two tables in the AWS-RDS: the city parcels original table and the table containing the selected attributes from the original parcels table and the attributes and records scrapped from the tax system website. Figure 1 illustrates the process just described.

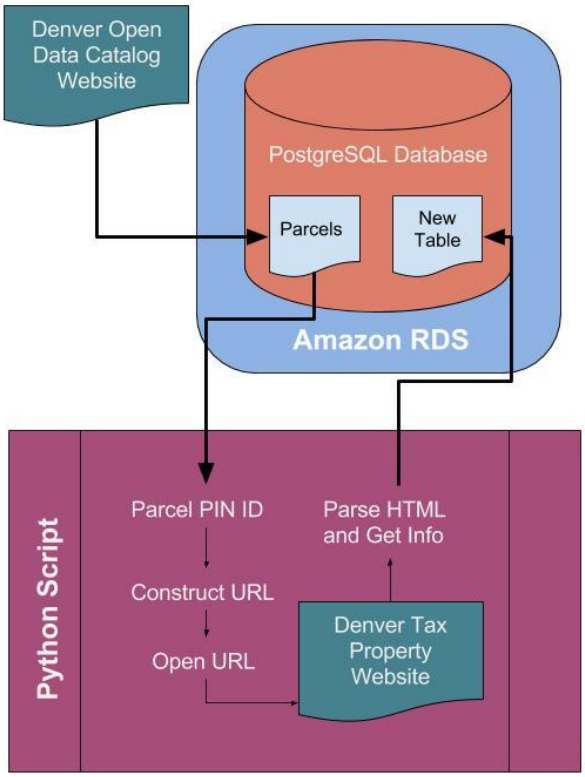

Figure 1. An overview of our methodology.

4. RESULTS

The test of matching of records between the parcels table and tax records table was successful. We tested scrapping diverse attributes from the tax system website and joining them to selected attributes and records from the city parcels table to create custom tables. Performance wise we found that the biggest constraint was how fast the tax system website could be accessed to retrieve the HTML code rendering the page corresponding to each parcel record. In our first tests, we let the Python script request the HTML of each page without delay, this resulted to be problematic as the website was unable to process the requests fast enough. After two hours of processing we cancelled the process. We found that adding as part of the script a small delay of 5 second between requests was enough to avoid this issue.

The information in the custom tables resulting from the processes described above are stored in the AWS-RDS and can be accessed by any software or GIS system that can connect to a database management system in a network. This would allow end users to easily join the new custom tables with any other information that have has a common unique identifier (such as the PIN number). For example, QGIS offers all required drivers to connect to PostgreSQL hosted locally or through a network.

Our Python script consisted of approximately 45 lines of code, most of them are simple adaptations of code found in well-known and well-documented free code libraries. The process to set-up a PostgreSQL database in the Cloud, such as in AWS as done here, has becomes much more user friendly and costs continue to decrease. Most Cloud services offer free 'tiers', our AWS RDS was in fact one of such free tiers, that allow for learning and also prototyping.

The FOSS, methods, and Cloud infrastructure here described are only one of several no-cost or low-cost options for combining information from Open Data repositories and made them broadly available through the Internet. Our focus was to find simple and reliable software and methods that are easy to learn and implement by users with basic technological skills.