Globeloc

What

globeloc is a data pipeline to create a global dataframe like object. This lets you put geospatial data into an array and perform computations with it. Therefore, this array can then be used to compute on a global scale removing the need to run geospatial geometric queries and instead using indexes.

Examples:

1. Find me all grocery stores within x meters of a point?
2. How many schools are there in the state of California?
3. What is the average number of schools per state per capita in the USA?
4. Use as an input into a location selection algorithm
5. Find all areas that have a elevation change less than 3%
6. Create a road graph from adjacency list
7. Calculate the nearest neighbor
8. Cluster data

This data is created such that it can be used in the same form as a pandas/R/spark dataframe

How

Let us take the entire set of lat lon coordinates (-180,-90) --> (180,90). This represents an infinite number of points. However, let's say we round every point to four (4) decimal places this constrains the possible number of coordinate pairs to an array of 3600000 x 1800000. Therefore 6.48 trillion, a number which is usable in the "big data" world of processing. This means that we still have "accuracy" to within around 10 meters. The actual number of items per 10 meters will vary and need to be computed by the user using standard haversine calculations.

Algorithm

```
# shift coordinate
# to non negative space
# multiply by accuracy factor
# round to nearest full number
s_lat = round((lat + 90) * 10000), 0)
s_lon = round((lon + 180) * 10000), 0)
```

```
# create empty array with the number of columns for the things that will be counted
# create empty rows for each latitude
# we will need a array for each vertical longitudinal slice

slice = [
        [‘ix’, ‘houses’, ‘churches’, ‘addresses’, ...]
        [0,4,5,6,7,8,9],
        ...]

# We could in theory iterate over all coordinate using this loop
# for j in range(3600000) # where j represents the column index
#       for i in range(1800000) # where i represents the row index

# Adding data to the array can be don’t by incrementing/adding the value of the cell i,j
# first iterate over all the data points and process the coordinates
# whilst iterating add the values together

data = {“name”: “houses”, “value”: 1, “coord”: (1235678, 300000)}
slice[data[“coord”][0], data[“coord”][1]] += data[“value”]
```

Storage

The array for each vertical latitude slice would be written to a ORC or other file format that is supported by a fast array processing software, blazingsql or bigquery or spark, etc...

We use vertical slices of files as we want to be able to filter how much data we load. These softwares out of the box provide a way to dynamically load a subset of rows and files.

https://docs.blazingdb.com/docs/text-files

Usage

- Users will be able to index into the array just like a regular dataframe
- Users will be able to add their own data
- Users will be able to submit query polygons (we create a array of [1,1,1,1,1] and multiply)
- Users will be able to get get metrics about the data just like the “describe” method in pandas

Monetization

The access to the data would be through an API, where you could pass your pandas like query. Users would pay a monthly subscription fee and then have a cap on the number of transactions.

Next steps

1. Build a jupyter notebook that has the data processing,storage,loading and query functions.
2. Use a subset of the osm data to create a data set (http://download.geofabrik.de/north-america/us/florida.html)
3. Test out a create some visualizations from the queries
4. Map out a plan to add the following:
    a. coverage of the USA
    b. elevation data
    c. address/population data
    d. wind and weather data
5. Create a api mockup to enable Python and js access
6. Develop a Python library to enable usage
   e.g. A python library like pandas
   g_df = globeloc.connect()
   g_df[10][100:120][‘hotel’]