Scaling for NYC Datasets

Report by: In-Memory Aggregators
Suyash Soniminde (sys8910)
Syed Ahmad Taqi (st4324)
Viha Gupta (vg2237)

Describe how you would run your approach on all NYC Open Data datasets?

There are many different aspects to look at if we want to run our approach on all of NYC OPEN datasets.

1. Identifying similar columns:
   The first challenge is to identify on which column to apply which cleaning techniques. For this we can indeed bucket the columns into different types, in our case the buckets are Boroughs, Dates, Titles, and Numerical columns describing hours or salary amounts.
   But in each dataset the columns are named differently, so basic string comparison or hard coding of columns name to type would not help here due to the massive scale.
   Hence we can do the following-
   a. Use similarity analysis to find if column names are the same. We should have a high similarity threshold in this case as even very similar sounding column names can represent different kinds of data.
   b. On top of the above, we should also check the exact datatype of the column values by sampling random portions. A profiler like one used in OpenClean is something which would work nicely here.
2. Improving the memory footprint of the scripts:
   In our particular case, since we are processing only about 14 datasets, we took the liberty to use PySpark Dataframes with all the columns intact.
   To scale this to entire NYC Open Datasets, which has datasets with varying sizes we can do a few things-
   a. Parse the dataset and use RDDs with only the relevant columns included. This way the executors will not run out of memory in case the dataset is large.

3. Making the cleaning methods generic:
   Make the methods to clean each of the column types as generic as possible so that they are relevant to all the datasets. This is the most challenging part of

cleaning since the same column type may mean something different in each dataset.
A few improvements to our approach are as follows:

a. For Borough names also consider the case of Typos and fix them using similarity analysis.
b. Perform analysis on the values of columns to find what kind of checks should be performed to remove outliers. Removal of outliers should be dynamic and the range or values should not be hardcoded in the script.
c. Different datasets have different datetime formats stored as strings. A better utility or library which can parse such formats and convert such columns to a standardized format should be used. If required we can explore creating a method from scratch.