# CS-GY 6513 Big Data Fall 2019 Final Project - NYC Open Data Profiling, Quality, and Analysis

Wenxuan Ma; Yuchuan Huang; Xiahao Zhang
New York University, Tandon School of Engineering

## Introduction

In this project, we derived metadata of 1900 datasets from NYC Open Data using Python, Spark and Pandas. We also extracted more detailed information from selected columns and used different methods to predict their semantic types. In addition, we analyzed the results with the help of applications such as Jupyter Notebook.

## General Profiling

**Architecture**

- The datasets are grouped by sizes (using shell scripts)
  - Small (<50MB), 1825
    - Use small datasets to test logic and correctness
  - Medium(50-100MB), 17
    - Use medium datasets to evaluation performance improvement
  - Large(100MB-1GB), 49
    - Divide spark-submit work evenly to each team member
  - Extra large(>1GB), 9
    - To prevent unforeseen errors and to save time from minor modifications, we store temporary output information after each column is done processing

**First Approach**

Use RDD

- Extract the header information
- Use map, filter, distinct, count, reduceByKey, sortBy, etc. to extract the number of non-empty cells, empty-cells, distinct values, and top-5 frequent values
- Use type convert to identify if a value is an integer or real(float)
- Use python's dateutil.parser library to identify datetime values
- If a value doesn't belong to integer, real, or datetime, identify it as a text(string) value.

**Improvements**

- Use Pandas dataframe
- Read the dataset column by column to reduce the memory cost (for large data sets)
- Avoid Python loops, use more Pandas's and NumPy's build-in functions such as max, min, mean, std, head, tail for optimization
  - Also reduce code complexity
- Avoid the use of Python libraries (or even Pandas or NumPy libraries)
  - Try to write our own map functions when possible
- Use Linux Shell scripts and screen command to help create multiple threads
- Evenly split the large datasets to each person, run separately and combine the results afterwards

**Results**

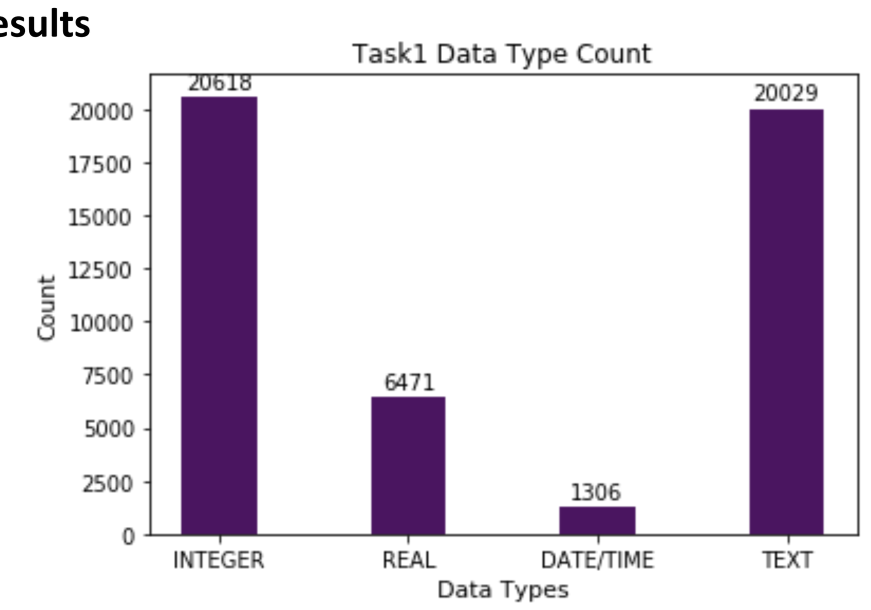



**Table.** Most Common Types - Frequent Item Sets

| Support | Itemsets |
|---|---|
| 0.521658 | (INTEGER) |
| 0.506755 | (TEXT) |
| 0.163723 | (REAL) |
| 0.140598 | (TEXT, INTEGER) |
| 0.093892 | (REAL, INTEGER) |
| 0.037091 | (TEXT, REAL) |
| 0.033043 | (DATE/TIME) |
| 0.019507 | (TEXT, REAL, INTEGER) |

## General Profiling (continued)

**Performance Improvement**

Small datasets(for all 1825 sets): 3 days to no more than 6 hours
Medium dataset(for one datasets): 3 hours each to 5 minutes each
Large dataset(for one datasets): Half a day each to 30 minutes each
Extra-Large dataset (for one datasets): After improvement, about 4 – 6 hours each

## Semantic Profiling

**Methodologies**

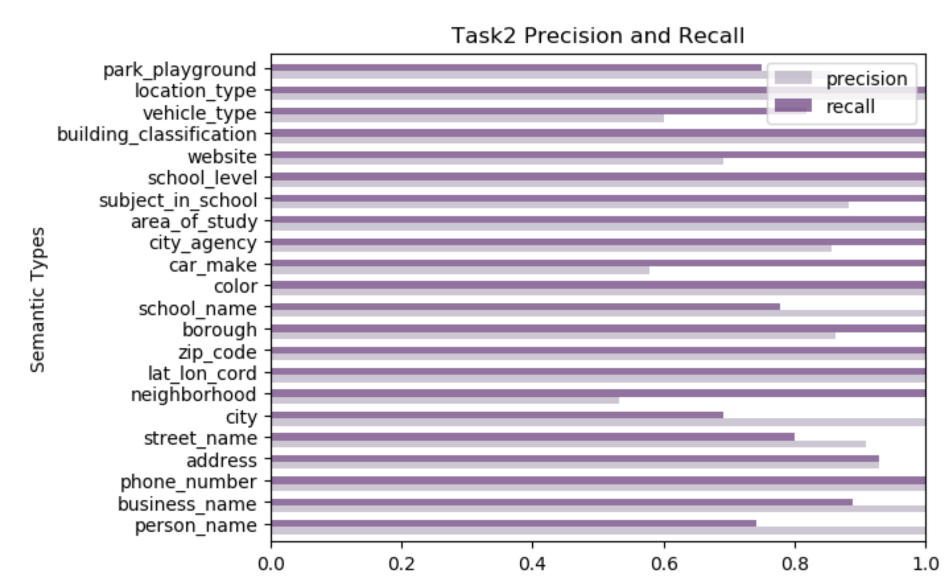
- Data Pre-Processing
  - Manually label all columns
  - One column may have multiple labels
  - Store the labels in a .csv file
  - Extract all distinct values that occur more than a given threshold of times, and group them by labels
  - Extract top 5 frequent values from the metadata json file derived in task1
- Prediction
  - Column names
  - Regular expression
  - Frequent Values (Fast)
    - Works well when there are dominant words in a semantic type
    - Likely to generate false positive
  - Search from known values (Safe, but slow)
    - Best solution if a semantic type has a limited value domain. Such as 'subject_in_school'.
  - External Libraries

**Challenges**

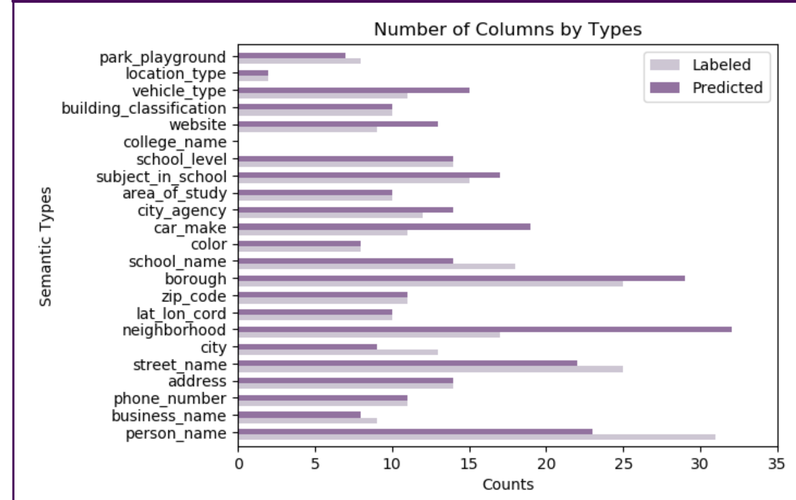
- Unclear Semantic Value
- Short value
- A value can have multiple semantic types
  - Brooklyn can be both 'borough' and 'city'
- Outliers

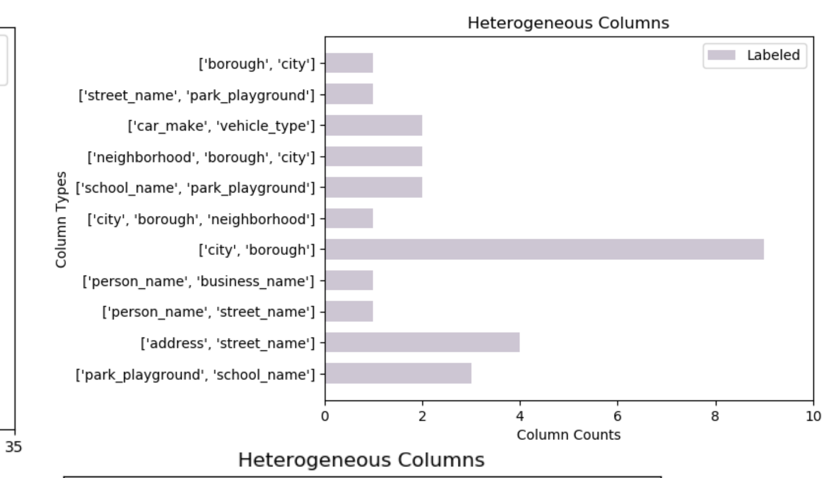
## Semantic Profiling: Results

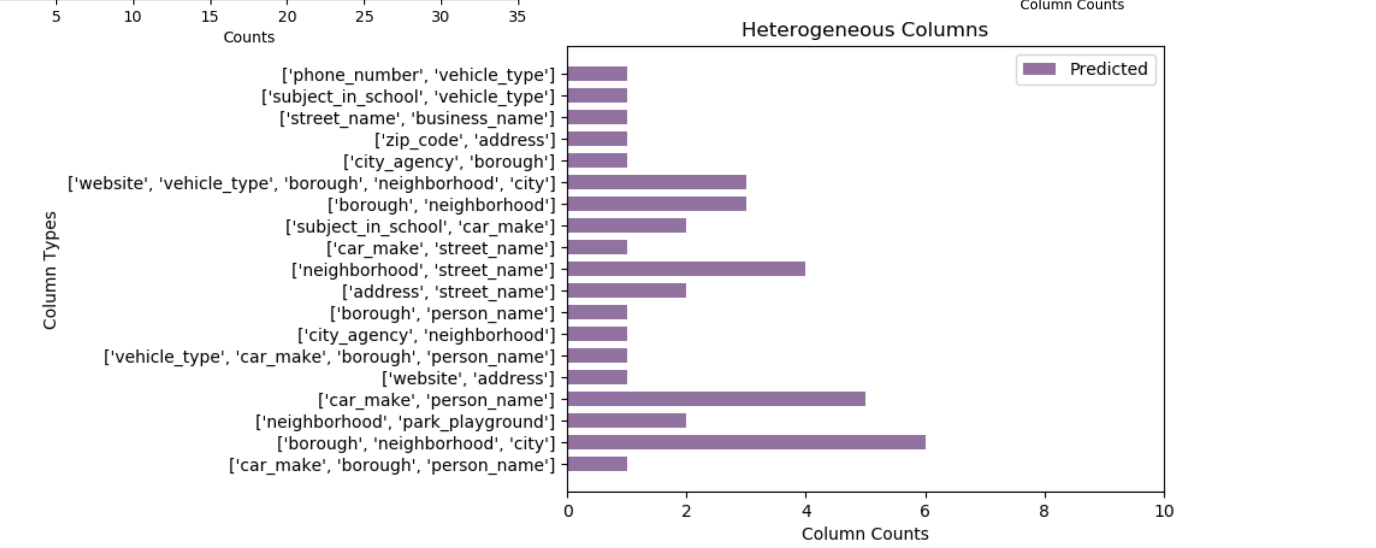
| Semantic Type | Precision | Recall | F1-Score |
|---|---|---|---|
| person_name | 1.000000 | 0.741935 | 0.851852 |
| business_name | 1.000000 | 0.888889 | 0.941176 |
| phone_number | 1.000000 | 1.000000 | 1.000000 |
| address | 0.928571 | 0.928571 | 0.928571 |
| street_name | 0.909091 | 0.800000 | 0.851064 |
| city | 1.000000 | 0.692308 | 0.818182 |
| neighborhood | 0.531250 | 1.000000 | 0.693878 |
| lat_lon_cord | 1.000000 | 1.000000 | 1.000000 |
| zip_code | 1.000000 | 1.000000 | 1.000000 |
| borough | 0.862069 | 1.000000 | 0.925926 |
| school_name | 1.000000 | 0.777778 | 0.875000 |
| color | 1.000000 | 1.000000 | 1.000000 |
| car_make | 0.578947 | 1.000000 | 0.733333 |
| city_agency | 0.857143 | 1.000000 | 0.923077 |
| area_of_study | 1.000000 | 1.000000 | 1.000000 |
| subject_in_school | 0.882353 | 1.000000 | 0.937500 |
| school_level | 1.000000 | 1.000000 | 1.000000 |
| website | 0.692308 | 1.000000 | 0.818182 |
| building_classification | 1.000000 | 1.000000 | 1.000000 |
| vehicle_type | 0.600000 | 0.818182 | 0.692308 |
| location_type | 1.000000 | 1.000000 | 1.000000 |
| park_playground | 0.857143 | 0.750000 | 0.800000 |




## Semantic Profiling: Results (continued)




## Candidate Columns being Keys of a Table

- Naïve Method
  - Select the columns which have N distinct values (N is the number of rows)
  - Can find such columns in more than half datasets
- Brute Force Method
  - Try all possible combinations of columns (only applies to small datasets)
- Improvement(apply to larger datasets)
  - Randomization
  - Sampling
  - Pruning
- Duplicate Rows: Remove or Not

## Data Analysis: Null Values and Outliers

**Null values**

- Finding null values: A combination of observation and auto-detection
- Observation: N/A, unknown, not applicable, repeated short values('*', '-', 's', etc.)
- Use libraries(like Pandas, NumPy) to help detect and analyze

**Outliers**

- Detect high frequency outliers (utilize task1 results)
- Distance-based outliers

**Dealing with Null and Outliers**

- Deleting the row / Replace the value with nan
- Use mean to replace the value

## Summary

During the process of profiling and analyzing the data, we explored, experimented and gained experience with a variety of tools, technologies and methodologies to deal with Big Data. There is also plenty of work that can be done in the future, like applying our methods to other datasets and try to improve the methods with new results we get.

## Contact

Email: wm1065@nyu.edu; yh2834@nyu.edu; xz2456@nyu.edu
GitHub Repository: https://github.com/wm1065/Big-Data-Final-Project
Results HDFS directory: /user/yh2834/2019-BigDataResults/

## References

1 Mining of massive datasets [Rajaraman, A., Leskovec, J., Ullman, J. 2012]
2 Profiling relational data: a survey. [Abedjan et al., VLDB 2015]
3 Pandas: a Foundational Python Library for Data Analysis and Statistics [W. McKinney, scipy2010]
4 Algorithms for Mining Distance-Based Outliers in Large Datasets [Knorr and Ng, VLDB 1998]