Data Profiling and Analysis of NYC Open Data

Nitin Anand Patil
nap493@nyu.edu
New York University

Samarth Shashikant Hiremath
ssh508@nyu.edu
New York University

Tanya Sah
ts3813@nyu.edu
New York University

ABSTRACT

Data processing or analysis cannot happen without having knowledge about the quality and reliability of data. Data has to be reviewed and the structure needs to be understood before we proceed to analysis. This makes Data Profiling a crucial stage in the data lifecycle. In this project, we profiled 1900 datasets from NYC Open Data for both generic and semantic properties. After profiling the data, we noticed that most of the columns are composed of heterogeneous data types and there are many columns with null values. Some basic analysis of the generic profiles of all the datasets gave us information about the data types which frequently co-occur in columns — Integer and Text data types occur together most frequently. With the data profiles available, we could clean the data and use it for further analysis. We considered the 311 Service Requests Dataset to analyse certain borough-wise and agency-wise trends in complaints. We did find multiple data quality issues in the dataset, but were all resolved and cleaned. As quite expected, noise turned out to be the top complaint type in all the five boroughs and Brooklyn accounts for the highest count of 311 Service Requests. It is also interesting that NYPD has the best non-emergency complaint resolution time.

INTRODUCTION

Each day large amounts of data pertaining to different categories is collected from different sources and stored by different organizations. But there is no easy way to tell whether the data collected is complete or accurate. Data quality issues often leads to bad business decisions which may incur huge losses to the organization. The data quality issues may persist due to incomplete values or null values in the dataset, formatting issues, incorrect values which altogether hinder the quality of data. For fixing the data we need to know about the data.

Data profiling is used to examine the data in order to gather statistical analysis over the data which can be valuable in improving the data. The following analysis is performed as part of Data Profiling:

- Completeness: Are all the values present in your data or are there any null values?
- Range: Finding the maximum, minimum and mean which helps in discovering any outliers.
- Uniqueness: The number of distinct values for each column.
- Pattern analysis: Different formats or data types for each column and statistical analysis over each data type.

Semantic profiling on the other hand gives a deeper understanding of data by establishing a connection between the real world and the data. Finding the semantic type for data is very important for tasks like schema matching, data discovery and automated data cleaning. Most of the semantic types can be detected by using regular expressions or by doing a dictionary lookup for the respective semantic type. But in few cases wherein the data includes values like the Person name, Organization name, Agency name, it is difficult to accurately classify the data into the respective semantic types.

Data profiling and Semantic profiling on very large data is usually very time consuming and requires a lot of resources in terms of memory, disk space and CPU. We use a Big data infrastructure, Spark by Apache which is a unified analytics engine for large scale data processing[1]. Spark includes many libraries including support for SQL queries and machine learning tools. Since Spark RDDs use lazy evaluation i.e execution will not take place till action is called, the required action can be called upon a small subset of data by performing a series of filter and aggregate functions. Spark also provides support for using udf (user defined functions) over data frames which helps in improving the performance of the code.

1 Generic Profiling

1.1 Methods

1.1.1 Single-column profiling

We analyze each column and compute the number of non-empty cells, empty cells, number of distinct values and frequent values for that column. Additionally, we compute the data types like Integer (Long), Real, Date/Time and Text, for each column wherein a column may have more than one data types. For each data type, we compute the number of values, maximum, minimum and average values.

1.1.2 Classifying data types

- Integer (Long)
  In order to check whether a column contains integer values we use regular expression that looks for one or more occurrences of digits and zero or one occurrence of the minus (-) or plus (+) sign. We use rdds in order to store the integer values and compute the maximum value, minimum value, mean and standard deviation for Integer values in a particular column by using the built-in methods.
- Real
  In order to check whether a column contains real values we use regular expression that looks for one or more occurrences of digits, a decimal point and zero or one occurrence of the minus (-) or plus (+) sign. We use rdds in order to store the integer values and compute the maximum value, minimum value, mean and standard deviation for Integer values in a particular column by using the built-in methods.
- Date/Time
  We use parser method from python's dateutil library to check whether the column has date or time values. It handles all formats of date and time formats in addition to incomplete month (Apr, Mar) or day (Mon, Tue) values. The values are stored as strings in the RDD and each value is mapped to its equivalent date-time format using the parse function. We compute the maximum value, minimum value and count for Date/Time values for each column.
- Text
  The values in a column are classified as text if they are not classified as Integer, Real or Date/Time type. The text values are stored in RDD and mapped to the text size in order to compute the top 5 longest values and top 5 shortest values. We also compute the average length for text values for that column.

1.2 Challenges

- The main challenge we faced in data profiling was that a particular column would have multiple datatypes and dataframes do not allow storing different datatypes in a column and were mostly stored as strings. We had to maintain a separate RDD for each data type with the values converted to their respective types in order to get the maximum and minimum values.
- Few datasets had columns with values for date or month in unconventional format like abbreviations for months and days, which couldn't have been handled by regular expression or python's datetime formatter. The parser method from dateutil was able to handle such values in order to classify correctly as Date/Time.
- In order to calculate the longest and shortest values for type Text, the values had to be mapped with their length and sorted.
- Certain column names have special characters like dot (.), slash (/) which made the queries over the dataframe for that column to fail as the query was not able to resolve the column name.
- Calling actions like collect, distinct on dataframe and RDD, very often would slow down the computation process when used on large datasets and would significantly affect the efficiency of the code.

1.3 Optimizations

- In order to speed up the code, we remove all the null values from the columns before performing any actions.
- We use RDD to map relevant data and aggregate the results by using Map and Reduce methods.

- The parser method from dateutil classifies any 2 or 3 comma separated values as date/time though it is not. This was fixed using a regular expression to check such occurrences and handle accordingly.

1.4 Data Quality Issues

- Many columns are heterogeneous i.e. columns have multiple datatypes.
- Few columns have missing or null values.
- Few columns use date or time formats which are not usually used to represent Date or Time
- The columns have outliers like city names present as borough names or the other way round.
- Few columns with text have lot of unnecessary white spaces leading to increased text length and hence consuming more space.

1.5 Analysis:

The results of data profiling revealed that Integer and Text data types were the most commonly occurring data types. Most of the columns have null values with few columns having majority of null values. Further analysis on the results of data profiling showed that many columns were heterogeneous columns with two or more data types present in them. The frequent itemset analysis reveals that Integer and Text values are the commonly found co-occurring data types.

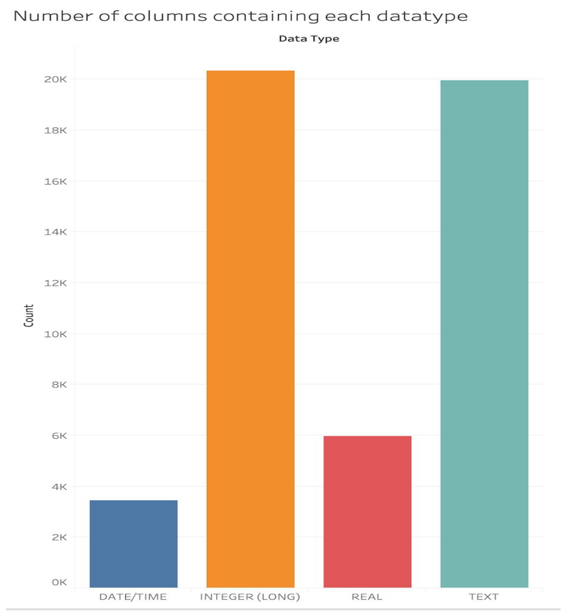

Fig 1.1: Count for each data type

Data Type1	Data Type 2	Count
TEXT	INTEGER (LONG)	3177
REAL	INTEGER (LONG)	1563
TEXT	DATE/TIME	1067
TEXT	REAL	783
INTEGER (LONG)	DATE/TIME	615

Table 1.1: 2- Frequent Itemsets

Data Type1	Data Type 2	Data Type 3	Count
TEXT	INTEGER (LONG)	DATE/TIME	588
TEXT	REAL	INTEGER (LONG)	585

Table 1.2: 3- Frequent Itemsets

2 Semantic Profiling

Semantic Analysis plays a vital role in situations of deriving information from unstructured and unlabeled data. It helps organization fetch meaningful insights from their data warehouse and utilize it for their economic growth. With the emergence of the Internet of Things age the volume of data has increased manifolds and along with it the randomness of the data. Thus, the need for semantic analysis has increased. In this task we explored various ways to extract semantic information of random data columns from numerous data files.

The 270 columns that were provided to us consisted of heterogeneous values and were of 23 different types. The different strategies that we employed for the semantic analysis can be categorized into the following types:

- Regex Pattern Match
- Dictionary Lookup
- Named Entity Recognition
- Soundex Algorithm
- Jaccard Similarity

2.1 Regex Pattern Match

Regex is one of the oldest and widely used approach for tagging text data with a fixed structural format. It gives quick and precise result in case of string searching. In our problem dataset we have Phone Number, Websites, Zip Code and lat_lon_cord which have a fixed format of representation and thus were easily classified using the Regex pattern match.

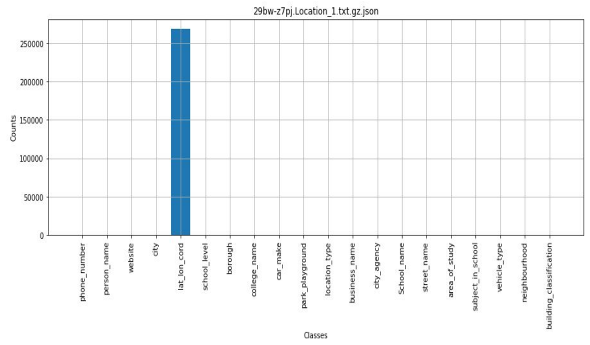

Figure 2.1: lat_long_cord classification using regex

2.2 Dictionary Lookup

In dictionary lookup we search for the words to be tagged in a pre-compiled dictionary. If the words belong to that dictionary than they can easily be classified to be of that group. In our case we used Dictionary lookup for the tags which had no structural consistency and consisted of random strings. This approach has the drawback that it requires data to be coalesced for creating the dictionary. *Used for car make, city agency, borough, school levels etc.*

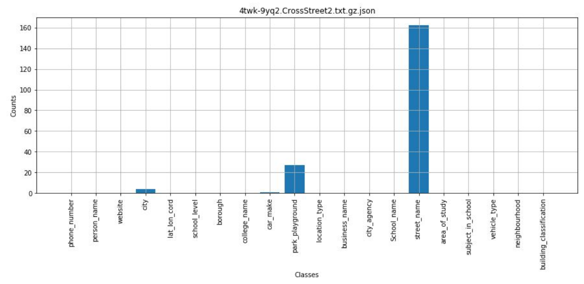

Figure 2.2: street_name classification using dictionary lookup

2.3 Named Entity Recognition

GloVe : To tag Person names and business names we first tried Global Vectors Representations for Words hoping to find any relation between the vectors. The trial was based on the hypothesis that vector distance between a person's name and person will be small. We tried by finding the similarity of a person's name and ‘person’ but the cosine similarity turned out to be very low around 0.29 and had to discard this approach for our dataset.

We then searched for different machine learning algorithms that could give optimal results for tagging named entity, One of the ongoing project ‘Sherlock’ uses Deep learning algorithm for classifying the named entity sets. We thus then searched for the research areas where such classification is studied which is Named Entity Recognition (NER). NER is a class of terms which is used to tag data words into predefined categories. Certain categories like Person Names and Business Names are difficult to classify using Regex Pattern and Dictionary Lookup. The dictionary for person names will have billions of records and hence not a proper approach for dealing with such tag classes. We explored the research area of Named Entity classification and realized that is used to with textual data and utilized the POS tagging in the texts to determine the semantic class of a tag.

To analyze the performance of the NER library on our column dataset which does not have any semantics attached with it we used Stanford NER library to tag the person's name and business name. Stanford NER library gave satisfactory results in comparison to the dictionary lookup using the baby names.

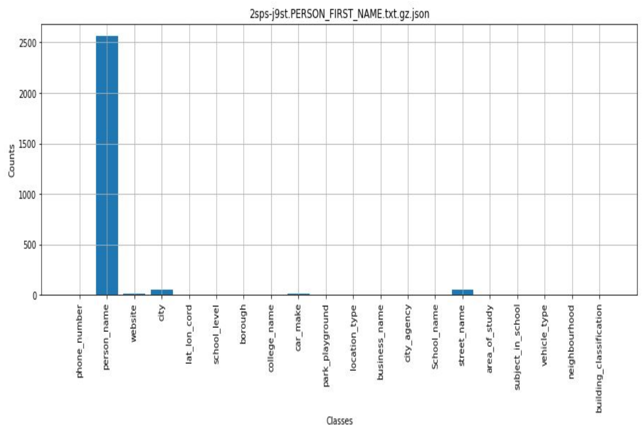

Figure 2.3: person_name classification using Stanford lib

2.4 Soundex Algorithm

Soundex is a phonetic algorithm which is used to group together the misspelled words on the basis of their phonetic sounds Analyzing the NYCColumn data we saw that there are certain columns like ‘Color’ which has a very less of accurate data and more of misspelled words. Ex. Yelloww instead of yellow. Soundex library is quite efficient in classifying such datasets.

Issues faced

- Soundex algorithm being phonetic when applied to datasets apart from colour started tagging other categories as colors. Tackled this issue by using soundex only for the colors dataset.
- Heterogeneous data columns: Heterogeneous columns caused issues in deciding the precise semantics of the dataset.
- Misleading information in the name of datafile. Ex. ‘sxmw-f24h.Park_Facility_Name.txt’ does not contains park name it contains school names
- Classifying Non-American person names: Stanford NER library was not able to tag certain Non American names like Sukhwinder which is an Indian name but the library classified it under others.
- Finding dataset for the dictionary lookup: Many datasets are not available online. Even if they are we need to filter out desired columns from the online available dataset. This is a time consuming task and adds overhead to the process
- NULL values in the columns. Removing Null values is important as it interferes in calculating join of the dataset.
- There are certain mis- tagged files which are creating false negatives. Ex dg92-zbpx.VendorAddress.txt.gz, this file only contains the street names still has been tagged as address.
- Data Format Issue: The generic format followed in files is the column values followed by its count. There are certain files deviating from this structures and have 3 or more columns in the dataset. ‘tg4x-b46p.ZipCode_s_.txt’

2.4 Analysis

The semantic analysis gave us an accuracy of 61.68% and Precision of 0.5 and recall 0.274 for the semantic profiling of the columns.

The results obtained through the semantic analysis showed that there were cases of false negatives than false positives.

- NYC boroughs are being classified as cities as the NYC boroughs are being also present in the city dataset.
- Neighbourhood being classified as Park_playgroud. In certain datasets the park_playground has neighbourhood data in it. This is making those files being classified as park_playground

3 Data Analysis

3.1 Problem Definition

Identify the three most frequent 311 complaint types by borough. Are the same complaint types frequent in all five boroughs of the City? What would explain the differences? How does the distribution of complaints change over time for certain neighborhoods and how could this be explained?

3.2 Data

We have used the “311 Service Requests from 2010 to Present” dataset from NYCOpenData collection for analysis. This dataset contains all the 311 complaints from the year 2010 till the date of this project. We have information regarding complaint type, complaint description, agency to which the complaint was directed, resolution taken, incident address, borough and all the dates related to the complaint. The dataset also provides information about the latitude/longitude coordinates, parks and taxi company names as per relevance with the complaint type.

3.3 Methods

The data analysis approach we have adopted uses the following methods -

- Dataframe - A Spark DataFrame is a distributed collection of data organized into named columns.

Spark allows many transformations/operations over the dataframe to filter, group, or compute aggregates. Dataframes can be used with Spark SQL and can be built from structured data files or existing RDDs. In our case, we are constructing the dataframe from 311 Service Requests structured data.

- User-Defined Functions - In PySpark, we can define functions as per our requirements and then register this function to be used with Dataframe. This feature is provided by the udf function of the functions class in pyspark.sql package
- Counting using groupBy() - In any dataframe, we can group the data based one or more columns using the groupBy function. Following such an operation will be an aggregation operation to count the number of values that belong to each group. This is performed by applying the agg() and count() functions over the grouped dataframe.
- Ranking - After computing the count for each group in a dataframe, to obtain data about top N counts in each group or least N counts in each group, the dataframe needs to appended with a new column denoting the rank. To ensure that we rank the counts within each group, we need to first create a window partition over the particular group.

3.4 Challenges

- There are many rows in the dataset with an invalid complaint type. The complaint type is non-alphabet and is hence of no use in our analysis. Such rows had to removed from the dataframe. This could be achieved with the use of regular expression matching. We are retaining only complaint type values that start and end with an alphabet.
- The complaint type column had values with inconsistent case. They had to be converted to a common format (lowercase) before grouping them. This problem was resolved by applying an user-defined function to the entire column. For optimized processing, the user-defined function handled both the regex check explained in the above point and converting to lowercase.
- Borough column was not completely updated and had few rows as 'Unspecified'. For such rows, the borough had to be figured out from the City column value, if available. We addressed this issue by having a predefined list of the five boroughs of New York City and verifying if the Borough column value is present in the list.
- Columns that contained dates related to the complaint were of the mm/dd/yyyy date format. This became slightly challenging when we tried to sort the data according to date or when we tried to retrieve data for complaints which were closed after due date. We changed the date format for all such columns to yyyy/mm/dd format using and user-defined function

3.5 Findings/Observations

3.5.1 Top 3 complaint types in each borough

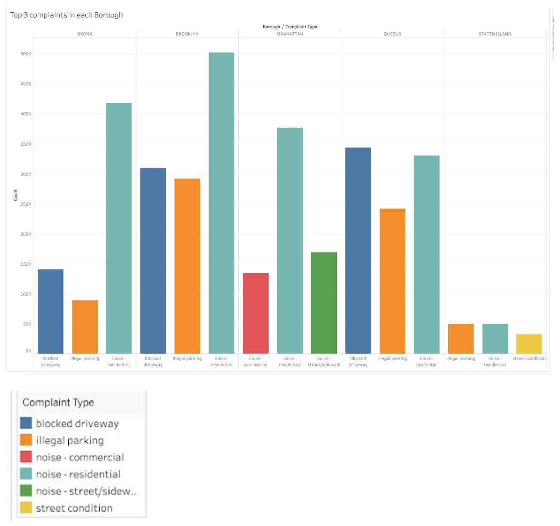

Fig 3.1: Top 3 complaints, borough-wise

From the above graph we can see certain expected results and can draw certain comparisons as listed below -

- Noise is one among the common major complaints in all boroughs. Brooklyn logs the highest complaints for residential noise. But that doesn't mean Brooklyn is noisier than Manhattan. When the population of these two boroughs is considered, approximately 22 complaints are lodged for every 100 people in Manhattan whereas, in Brooklyn, the number is 18 complaints for every 100 people.
- Another interesting observation (as we expected) is that the top 3 complaint types in

Manhattan are all related to Noise (Noise - residential, Noise - street, Noise - commercial)
- In Staten Island, complaints for illegal parking are higher than other boroughs

3.5.2 Top 3 agencies to which complaints are directed

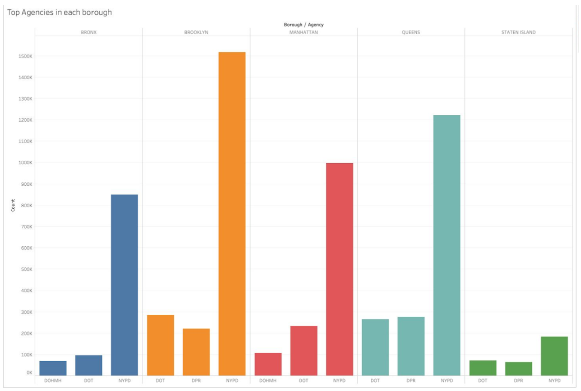

Fig 3.2: Top 3 agencies, borough-wise

- NYPD records the highest complaints in all the five boroughs. When compared with the population, we see that the complaint rate is highest in Manhattan (60 for every 100 people) and the lowest in Staten Island (38 for every 100 people)
- Complaints to Department of Health ranks among the top 3 only in Manhattan and Bronx. This could probably be because of the higher number of restaurants in Manhattan

3.5.3 Percentage of complaints that were resolved after due date

- Among the five boroughs, Manhattan is the one with higher percentage of complaints that could not be closed within estimated due date. This probably could be because of the nature of complaints since most of them noise related.
- Among the different agencies, the NYPD has the least percentage of complaints that crossed due date - 3%, which means NYPD has the best non-emergency resolution time. As per data, the DEP (Department of Environmental Protection) records highest percentage of missed due dates. But this number is a complete 100%, which gives rise to doubts whether the due date was not properly updated. Hence, ignoring the DEP, the highest percentage of missed due date is for EDC (Economic Development Corporation) - 87%

3.5.4 Distribution of complaints over time

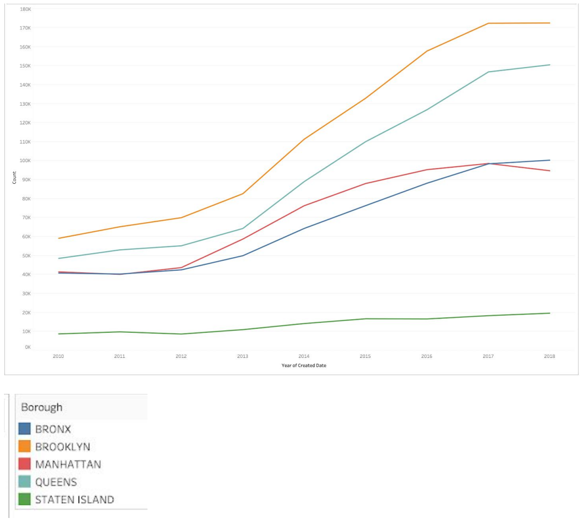

Fig 3.3: Complaint Rate vs Time

This graph clearly shows an increase in the complaint rate over the years. This could be connected with various reasons like population growth and more service requests being digital over the recent years. We can also observe that the rate of complaints has increased more rapidly for Brooklyn and Queens than Bronx and Manhattan. The complaint rate in Manhattan has decreased slightly over the last few years.

INDIVIDUAL CONTRIBUTION

- Samarth Shashikant Hiremath
  Worked on data profiling of 1900 datasets of NYC Open data using Single-column profiling method and analyzed the distribution of different data types over columns.
- Tanya Sah
  Worked on semantic profiling of the datasets
- Nitin Anand Patil
  Worked primarily on analysis of 311 Service Requests Dataset from NYCOpenData along with minor contributions in generic profiling and semantic profiling.

Code and Output
GitHub:
https://github.com/nitinz1994/Big_Data_Project
HDFS directory path:
/user/nap493/Big_Data_Project

CONCLUSION

Data profiling is a crucial stage before data analysis since it helps us understand the quality of the data and devise appropriate data cleaning strategies. It also gives us an idea about the number of columns that have null values, which can help identify key columns. When we deal with large datasets, distributed computing plays an important role and significantly increases computation speed.

REFERENCES

1. Apache Spark Official Documentation - https://spark.apache.org/docs/2.2.0/api/python/pyspark.html
2. Article on the use of Spark Dataframes - https://mapr.com/blog/using-apache-spark-dataframes-processing-tabular-data/
3. Wikipedia - https://en.wikipedia.org/wiki/Neighborhoods_in_New_York_City
4. Wikipedia - https://en.wikipedia.org/wiki/Boroughs_of_New_York_City
5. Official website of the city of New York - https://www1.nyc.gov/
6. Python dateutil documentation - https://dateutil.readthedocs.io/en/stable/parser.html
7. Profiling relational data: a survey. Abedjan et al., VLDB Journal 2015
8. Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. ACM SIGKDD International Conference on KnowledgeDiscovery & Data Mining (KDD '19), 2019.