Fall2019 Bigdata project - CoinFlippers

Wayne Young(yy3090)
Mina Lee(ml6543)
Sujay Lokesh(sl5202)

, ,

1 ABSTRACT

We live in an age where massive amounts of data is created every day. It is estimated that about 2.5 Quintilian bytes of data is created each day. Data is being generated without any thought. With all this data being created, it was thought that generating thoughts and conclusions from this massive amount data will help in being able to result in faster service, better prediction or even assist you better in your daily life. This gave rise to the study of Big Data. In this report, there will be a clear outline of the work done on NYC Open Data. This large collection of public data helps generate insights that can help the City Government serve the publiv better. By using this data, we can research conclusions and request resources where it is needed, and there is factual evidence that shows that it is required through analysis.

In Task 1, Metadata for NYCOpenData datasets is generated. This can be very helpful as it is possible that all NYCOpenData datasets have an adequate amount of metadata. This metadata includes the counts of cells, the datatypes and statistics related to the dataset.

In Task2, identification of the semantic types for each column in an NYCOpenData dataset was performed. The semantic types include Person name, Address, LAT/LON coordinates, Building Classification, School name, and so on.

In Task 3, Analysis on 311 data from 2010 to 2019 was done. Questions like "What are the most frequent complaints in Brooklyn?" and "How long on average does it take for a complaint in Manhattan take to get solved?" were solved. There was also some visualization done which can be used to show the distribution over time.
github repository:
`/achievermina/2019Fall_Bigdata_finalProject.git`
HDFS repository:
`/user/ml6543/2019Bigdata_coinFlippers/`

2 INTRODUCTION

The growing popularity of Big data analysis can be accredited to its convenience for processing large data sets. The Big data management system has rapidly developed from conventional information technologies. The explosion of information in the last decade has enabled the need for organizations and individuals to have the ability to process and save enormous amount of data. Interpreting data is becoming crucial for variety of needs. Businesses can rely on data analysis to pin point flaws and customer needs to achieve goals like increasing profit margin and reducing cost. For non-profit organizations, data interpretation is also important to help improve service qualities, and enrich the understanding of different situations.

This project addresses the benefits from big data infrastructure by operating multiple steps of data cycle and manipulating data for analysis. For our project, we are determined to profile and analyze New York Open Data sets. Three contributions are made. First, a variety of strategies for profiling and cleaning data has been achieved. Second, we proposed multiple algorithms to determine the semantic types for data sets. Lastly, the profiled data set is applied for unlocking the insights.

3 TASK1

The main purpose of task 1 is to profile generic data sets and generate metadata. Metadata is a general glance of the data and it gives us a rough understanding of its composition. An accurate description of the data can dramatically improve the data processing efficiency. However, there are many factors to take under consideration when it comes to generating metadata, including data quality issues, processing power and precision etc. The following sections will discuss our solutions and motivations behind them.[1]

3.1 Method

3.1.1 Software and libraries.

- requests==2.22.0
- pyspark=2.4.4
- pandas=0.25.1
- fuzzywuzzy=0.17.0
- plotly=4.4.1

'requests' are used for accessing data through REST APIs, since the network speed is much higher than disk I/O. 'pyspark and pandas' are used in combinations to ensure the ease of implementation and performance. 'fuzzywuzzy' is used for classifying similary strings using different algorithms. 'names datasets' is a training data set that can help us train a machine learning model which can identify names across multiple cultures and nations. Plotly is used to make the bar and scatter plots which have been shown in Task 3. (6)

There are some steps required to complete the metadata generation. The process for profiling a table is rather straight forward. However, multiple challenges and caveats are still presented.

3.1.2 Reading the data. There are mainly two ways to acquire large data sets in general: though networking or disk I/O. It is possible to combine both, for example, we considered to read all data into an in-memory database such as Redis to improve the performance. However, such solution was not adopted due to the time constraint and lack of resources (we do not have a machine to be used as a stable server). For our implementation, we store most of the data in memory and access them through panda data frames. There is a couple of data sets which we get through network to keep the data up to date and accurate.

3.1.3 *Profile the statistics for a single column.*

There are multiple ways to calculate the statistics for each column. The solution adopted was to use SparkSQL. SparkSQL is easy to implement and it has a decent performance because it can automatically distribute work to the cluster. Then we use the statistics library to aggregate the data we acquired from spark.

The steps are the following. We can start by reading the data set from a source, all data sets will be read into a dataframe, we can calculate statistics (max, min, unique count) of each column rather easily using SparkSQL (avg, count). Then we can find frequent item sets (count [column name] group by). After this we can simply check the data type and run statistics profiling specific to each value's data type.

3.1.4 *Identifying the Key columns.*

Key columns refer to the columns which are unique and can be a primary key. We simply check if the non-empty rows are unique and subtract it from a threshold to account for human errors. The value of the threshold turns out to be not important and it's usually a good estimation to just account for the uniqueness of the column

3.1.5 *Data type identification.*

In order to profile the statistics, we would have to profile for data types first. Since each data type would have different type of statistics. It would not make sense for a numeric type to have length property; dates and text should not have mean and standard deviation. Fortunately, it does not take an extremely advanced algorithm to determine the data type for a field. There are following types: alphabet, numeric, alphanumeric, date and text. The presence or absence of digits and alphabets can easily help us distinguish three of the data types. As for dates, we can use either a built in python function to convert a string to date data type, or we can use regular expressions for a more general profiling. We decided to used built in functions from python, since we have checked our data sets, and there only one format for date data. Therefore, we can use a fast function to deal with this type of data, and a more general regular expression is not needed under this circumstance. In the program, we retrieve all rows for each column and we run the algorithm described above against all of those rows.

3.2 Challenges

3.2.1 *Data quality issue.*

There are multiple data quality issues for the given data sets. As far as we know, a large portion of NYC Open Data are gathered through human input, and human errors are impossible to avoid. Even if there is no errors, we would still have to deal with different data presentations that is hard to be handled automatically by any program. For example, there are missing data for numerous columns, which are challenging when it comes to finding dependencies and composite keys due to the sheer volume of the data.

Typos and formatting issues are also a common issue when it comes to data quality. The issue does not only present in data itself, but also the data descriptions, such as table names and column names etc. For instance, we have encountered common names with percentage signs, and signs. On top of that, many of the columns can be numbers. Those column names can easily result in SQL syntax errors and unexpected termination of the program. Therefore, parsing of the columns is needed. We replace the column names and table names with legal characters that would be recognized by the compilers so we can perform operations in the data sets. For example, we had to clean the data and replace illegal characters and symbols with English or underscores to make column names queryable in spark SQL. One draw back of this solution is it can sometimes generate duplicate column names since different characters might be mapped to the same outcome. To resolve this issue, we keep a map of each column name and its number of occurrences. We append the number to the duplicate column names to ensure uniqueness.

Typos and mistakes in the data set can be frustrating to deal with, therefore we use fuzzywuzzy as the library to check for similarities. One solution that could be optimal is to use min hashing to group similar values together. However, we found that there is no significant improvement for performance and accuracy for a small cluster of machines for such an algorithm, we we would need significantly more memory to keep track of the similarities which often result in out of memory errors., especially for a shared cluster. We find that the best way to profile the data is thought estimation rather than counting.

3.2.2 *Data cleaning.*

It is natural to consider to clean the data sets for more accurate metadata generation. However, such an operation is not practical for this task for the most part. The number one most important precondition for performing data cleaning is domain knowledge of the data sets. In this case, we do not have any detailed understanding for all the data sets we come across with. Therefore, it is quite impossible to decide which data to delete or keep for a generic set of data. For our purpose, we discarded empty and duplicate values.

3.2.3 *Optimization.*

Running a data profiling task for a data set can take a long time. Different strategies such as utilizing spark map and reduce algorithm, reducing data size, and configuring cluster resources are employed during the implementation of this task.

Spark RDD and map reduce. To optimize the resources of spark, we try not to convert spark objects to python objects. Spark RDD is the advantageous data type to analyze the data using map and reduce strategy. Using the parallelism which Spark offers, we maximized the code efficiency.

Reduce data size. For some statistics, we used the reduced data with distinct values. For instance, searching the distinct values for the shortest/longest values of text data column and maximum/minimum value of date column was significantly reduce the running time instead of iterating duplicates.

Configure resources for the cluster. We set the configuration of how many cores and memory we allocate for each program instance. It turns out the most optimal configuration is 5 cores and 8gb of memory. This change resulted in 50 percent speed increase. Using panda dataframe. According to our test, Panda dataframe are much faster than Numpy and python list objects. We can usually expect a 30 percent speed increase from Panda dataframes.

3.3 Results

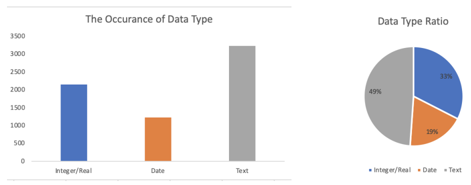


Figure 1: Data types in columns

We found out that some columns contain multiple data types. Using the frequent itemset approach we found out that integer and text co-occur frequently. About 30% of text columns shows the co occurrence with the text data values. Also, we can see the majority of the data type is text, and we can assume that it is because we consider the majority of columns such as ID or building numbers as a text.
By analyzing the statistics of the datasets, we found out many of the data columns have null values. For our statistics, we added null value when it is one of the most frequent values. From this analysis, we were able to determin how many columns were totally empty column and for data analysis in task3, we have removed these unnecessary columns for efficiency.

4 TASK2

Data profiling is the most challenging task so far. Our goal is to determine the semantic types for each column and the number of occurrences of each. The semantic type is defined row by row. This task requires us to match strings on a large scale. There are a number of ways to match strings, including EditDistance, Longest Common Sequence, LSH, Regular Expression, Machine Learning etc. We will explain our solutions and motivations for each circumstance. Different strategies will also be discussed.

4.1 Approach

There are about thirty different semantic types we wish to classify the types with. The variety of these types are significant. Therefore, there is no single strategy that can deal with all of them efficiently and accurately. We will discuss our strategies chosen for profiling different semantic types, why we chose them, and how well they work.

4.1.1 Short words matching.
For short words, we used Edit Distance to match two strings. This method is really useful for short words matching to detect the typos and minor mistakes. However, we found that the majority of the data sets are not likely to be short. Therefore, a more versatile set of methods are needed to be employed.

4.1.2 Non-short words matching.
To address the short comings mentioned above, we decided to use FuzzyWuzzy python library to match strings. Here are some of the the cases we need to deal with when profiling for semantics. A number of solutions are employed using Fuzzywuzzy library for partial matching and long string matching under different situations.
For example, we might get a good match for two pair of strings "New York City" and "New York Cite" using Edit Distance, but if we have "NewYork city" and "New York Cite" then edit distance will give us a much lower matching score. A problem the library addresses is inconsistent substrings. For a pair of strings "YANKEES" and "New York Yankkess", the library would give them a higher matching score despite the large difference in edit distance. It values the matching score more on the shorter part of the string rather than the longer part. For larger set of text, we can have more problems such as text is out of order, string volume is too high etc. Fuzzywuzzy allows us to easily tokenize the text to complete a set of operations needed. Since we do not have the needs to compare sets of documents, algorithms like min hashing and LSH are not chosen for this task
This method is quite useful for handling string matching for business names, area of study, general description and data similar to these which do not have strict format and have moderate lengths. The partial matching words well with a tight threshold value. It is common a

4.1.3 Regular Expression.
There are many fields of data which have strict formatting, string matching would be much easier in this case if we were to use regular expression. For instance, for numbers like phone numbers, URLs and zip code etc. It is worth nothing that we will encounter faulty data that will not match with the regular expressions specified. It is acceptable, however, considering we would delete the wrong data regardless.

4.1.4 Keyword Searching.
It is effective and easy to identify certain semantic types where the data can be predictable. For example, School names usually have words like "school", "University", "college", "high" and "institution" etc. Street names usually have keywords like "ST", "AVE" and might contain zip codes or city names. Using simple methods like this can drastically improve the performance of the profiling process and we can still have a good estimation with decent true positive rate.

4.2 Methodology

4.2.1 Sound Index.
Sound Index or SoundEx is a term used to describe the process of indexing names based on their phonetic spellings. SoundEx is a useful tool as it helps us in identifying any misspellings that may have occurred. For example, the word "Brooklyn", can be misspelt as "Brewklyn". We can understand that the the input is supposed to be "Brooklyn" but has been entered wrong due to Human Error. When writing algorithms which runs on massive datasets, the computer would not understand the true meaning and then interpret it as "Brewklyn". This will result in an erroneous result. Soundex can solve this issue by calculating a SoundEx hash value by using the first letter of the name and converting the consonants in the rest of the name to digits by using a simple lookup table. Vowels and duplicate encoded values are dropped, and the result is padded up to—or truncated down to—four characters.

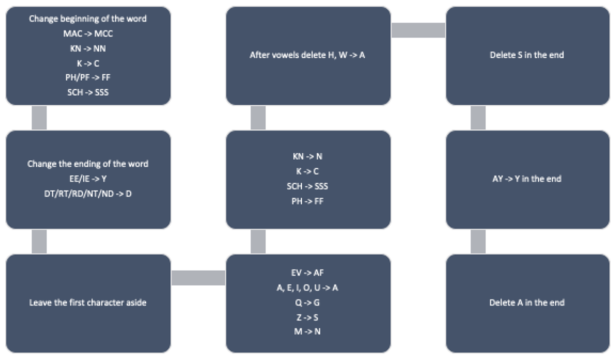


Figure 2: NYSIIS Algorithm

New York State Identification and Intelligence System Phonetic Code (NYSIIS).
After SoundEx, many other algorithms have been implemented. All of them process phonemes differently in an attempt to improve accuracy. For example, in the 1970s, the New York State Identification and Intelligence System (NYSIIS) algorithm was published by Robert L. Taft. NYSIIS was originally used by what is now the New York State Division of Criminal Justice Services to help identify people in their database.

4.2.2 *Heuristic Sampling method .*

Profiling the majority of the data set is a burden regardless of what resources we have. Therefore, Big data analysis often choose estimation over precision. Using statistics and probability, we found out the estimation does give us an acceptable result. To profile the large data set, it is reasonable to collect samples and determine the type of data. For this project, we sampled from distinct values of the original columns.

Reduce data size. We established that we do not need all data for profiling. We decided to randomly sample the data because we found out it is quite common for data which are entered closely within a time period to be similar. That means we can introduce bias to our data set if no sampling were used across the entire row. It is worth noting that we use distinct values for both methods whenever we can to reduce the data size. It is very common to have the size of most rows reduced by a significant percentage, sometimes more than 90 percent, when the data sets are distinct, since key columns are the minority for any data set.

Sampling.
Sampling allows users to create samples reflecting the whole dataset efficiently. There are new sampling methods developed recently for big data analysis such as Zig-Zag sampling[3]. we have tried different sampling methods to increase the speed and get diverse results. However, we couldn't find the big differences in speed or sampling accuracy with the project datasets. Therefore, we implemented uniform sampling which is already implemented in Apache Spark. Furthermore, we went through multiple iterations of testing to discover the accuracy of the sampling sizes and decided to use at most 1000 samples.

Decision threshold.
As there is no designated number for the decision threshold, we took the heuristic method. We look for the best threshold with our soundex approach by try and error. First, we manually selected 100 columns with diverse semantics to explore the accuracy of semantic check functions. Changing the thresholds of the semantic check functions, we have found that we need to adjust the threshold based on the semantic types. Some semantic types such as street name or website worked with high threshold. Generally, we set up the threshold with 50%.

4.2.3 *Data cleaning.*
The situation is similar to what we mentioned above for task 1. Data cleaning is possible, but the return of performing such a task is low compared to the effort that needs to be spent considering the extent of the variety of the data sets given. A regular and generic strategy is employed for this task. Null values are duplicated were eliminated.

4.3 Challenges

4.3.1 *Faulty Data.*
There exist more challenges when it comes to parsing faulty data than the process of analysing the data. Even though in English, there are different words to express the same meaning. Abbreviation and incomplete data was also one of the challenges we had to deal with. As we decided to use the soundex apporach, collecting the most potential values in each semantic type was needed. We gathered datasets for each semantic type and use them as a reference.

4.3.2 *Heterogeneous and diverse dimensionality.*
Different information collectors often uses their own protocols for data collections. This results in diverse representations of the same data.[2] For instance, in the school related dataset, there were same columns of interesting subjects from 1 to 6 and mostly from interest3 to 6 was null values. This was challenging because our goal was determining the semantics by columns and without the domain background about the whole dataset, it was hard to determine the correct semantic. We could rely on the column names instead and it seems reasonable, but for this project we avoid those methods in order to apply diverse algorithms we have learned from the class.

4.3.3 *Memory management.*
Low-level concepts like memory management is usually neglected for high-level programming frameworks and languages such as Pyspark. This is understandable since the memory is managed by garbage collectors. The memory structure is also complex due to its nature being distributed across multiple machines. There is little we can control as far as memory allocation goes, but we can still control the memory usage by knowing what the program does. At one point, we keep getting Out Of Memory exceptions for tables containing large a great number of columns. There are too few Memory profiling tools for python and they have limited functions and readability for our needs, therefore, the source of memory over usage was tracked through trail and errors by adding and deleting lines of code across the program. We found out the memory was

hogging by the information saved for all columns within a table. Our solution was to parallelize the data across multiple machines instead of using only a single machine. Details of such a solution is discussed in the following subsection.

4.4 Results

We approached with two different strategies. First, we used the different threshold for some semantic types, which could be determined easily by checking if the value contains certain words. For instance, if the value contains 'st' or 'avenue', it is recognized as an street name. For these semantic types, we increased the threshold. Otherwise, we used the lenient threshold we got from the heuristic approach. By mixing these two strategies, our outcome detects the semantics with high rate of precision. However, it often returns multiple labels. Even though the correct label was included in the label lists, the recall rate of our outcome was relatively low. The result from the final run was precision rate 72% and recall rate 35%.

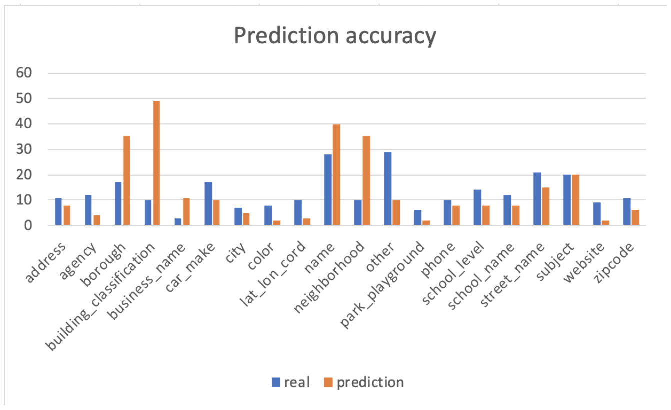


Figure 3: accuracy

Figure above shows that some semantic types have higher false-positive rate. Building classification, person name, and the neighborhood have detected more often than others even though the column was not related to those semantics. We assume that this is because some names and neighborhoods have similar words.

5 TASK3

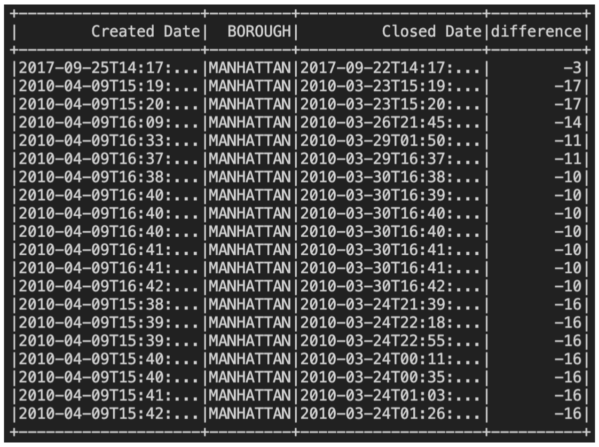
5.1 Introduction

311 is a non-emergency phone number that people can call in New York City to find information about services, make complaints, or report problems. We can gain a more profound understanding on things like the percentage of the calls for different areas, complaint type, who and when the complaints are made. Information extracted from such an analysis can be used to help allocate resources for dealing with certain area or certain types of complaint calls.

5.2 Data Cleaning

The 311 dataset obtainted from NYCOpenData can be termed as dirty. Data Cleaning or Data Cleansing is the process of detecting and correcting or removing corrupt or inaccurate records from a record set. Data Cleaning is important as incorrect data can lead to false conclusions and lead to investments in the wrong area. For example, if the city is looking to hire more employees to handle the increase in a specific complaint. If the data is not cleaned and the wrong complaint type had been selected then too many of the wrong resource will be hired and can lead to further problems in the City. We can say that the data is clean if :

- Accuracy
- Validity
- Completeness
- Consistency
- Uniformity



```
+--------------------+---------+--------------------+----------+
|        Created Date|  BOROUGH|         Closed Date|difference|
+--------------------+---------+--------------------+----------+
|2017-09-25T14:17:...|MANHATTAN|2017-09-22T14:17:...|        -3|
|2010-04-09T15:19:...|MANHATTAN|2010-03-23T15:19:...|       -17|
|2010-04-09T15:20:...|MANHATTAN|2010-03-23T15:20:...|       -17|
|2010-04-09T16:09:...|MANHATTAN|2010-03-26T21:45:...|       -14|
|2010-04-09T16:33:...|MANHATTAN|2010-03-29T01:50:...|       -11|
|2010-04-09T16:37:...|MANHATTAN|2010-03-29T16:37:...|       -11|
|2010-04-09T16:38:...|MANHATTAN|2010-03-30T16:38:...|       -10|
|2010-04-09T16:40:...|MANHATTAN|2010-03-30T16:39:...|       -10|
|2010-04-09T16:40:...|MANHATTAN|2010-03-30T16:40:...|       -10|
|2010-04-09T16:40:...|MANHATTAN|2010-03-30T16:40:...|       -10|
|2010-04-09T16:41:...|MANHATTAN|2010-03-30T16:41:...|       -10|
|2010-04-09T16:41:...|MANHATTAN|2010-03-30T16:41:...|       -10|
|2010-04-09T16:42:...|MANHATTAN|2010-03-30T16:42:...|       -10|
|2010-04-09T15:38:...|MANHATTAN|2010-03-24T21:39:...|       -16|
|2010-04-09T15:39:...|MANHATTAN|2010-03-24T22:18:...|       -16|
|2010-04-09T15:39:...|MANHATTAN|2010-03-24T22:55:...|       -16|
|2010-04-09T15:40:...|MANHATTAN|2010-03-24T00:11:...|       -16|
|2010-04-09T15:40:...|MANHATTAN|2010-03-24T00:35:...|       -16|
|2010-04-09T15:41:...|MANHATTAN|2010-03-24T01:03:...|       -16|
|2010-04-09T15:42:...|MANHATTAN|2010-03-24T01:26:...|       -16|
+--------------------+---------+--------------------+----------+
```

Figure 4: Need for Data Cleaning. Closed Date before Created Date

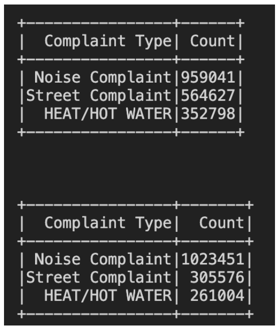
In 4, We can see that the difference between the Closed Date and the Created Date results in a negative number. This would mean that the the Complaint was resolved before the Complaint was made. This is obviously not possible and therefore brings the need for Data Cleaning.

5.3 Analysis

While calculating the Top 3 types of Complaints for each borough. We see that Noise Complaint is a common type of complaint among all the 5 Boroughs. This is shown in the figure 5. While calculating the average response time of 311, We can see that the trend of the average response time in the Brooklyn area. We can conclude from this that the average response time has reduced over time. This can be visualized in figure 6

Using Windowing functions, we were able to find the highest occurring complaint type per Borough per year. This is important as it can help us analyze trends and how it changes over time for a particular borough.

Another analysis performed here was identification of the number of complaints per person. The total count was taken and divided with the population of that borough. The population data was obtained at NYC.gov. Using the Plotly module, we were able to visualize the top 10 complaint types in New York City in figure 7.



```
+-----------------+------+
|  Complaint Type| Count|
+-----------------+------+
| Noise Complaint|959041|
|Street Complaint|564627|
|   HEAT/HOT WATER|352798|
+-----------------+------+


+-----------------+-------+
|  Complaint Type|  Count|
+-----------------+-------+
| Noise Complaint|1023451|
|Street Complaint| 305576|
|   HEAT/HOT WATER| 261004|
+-----------------+-------+
```

Figure 5: Top 3 Complaint Type per Borough

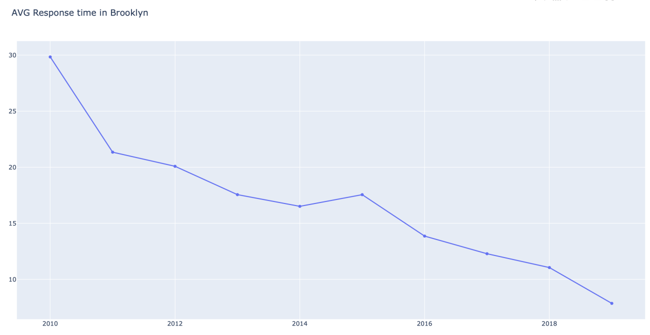


Figure 6: Avg. Response time in Brooklyn

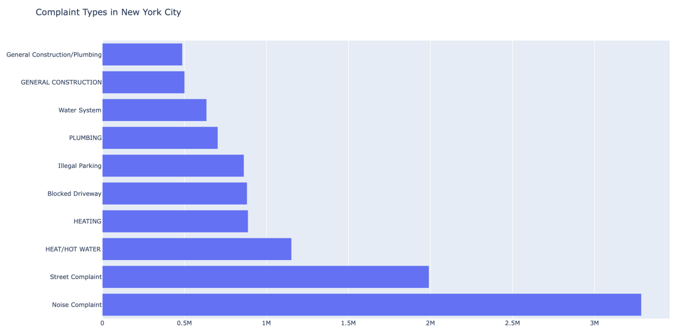


Figure 7: Number of Each Complaint Type

From this we can see that more needs to be done to prevent and reduce the number of Noise Complaints. The use of these techniques helps show the city government where more resources need to be allocated.

Lastly, while exploring the status of complaints. It is found that most complaints have been closed. Although, there are a considerable amount of complaints which are neither open or closed. Closed has been omitted from figure 8 as it represents about 80 percent of the data set. The remaining 20 percent shown here tells us that there are more options other than Closed and Open. This helps people understand that their maybe some Complaints getting lost.

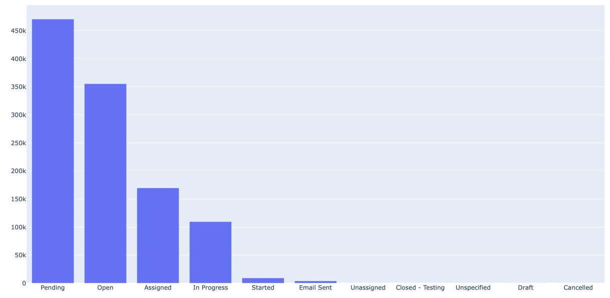


Figure 8: Satus of Complaints without Closed

5.4 Methodology

To Calculate the Frequent Complaint types:

(1) Aggregated by 'Borough' and 'Complaint Type' to get the total count of complaints registered for each combination of 'Borough' and 'Complaint Type'
(2) The resulting table is sorted by Count of Complaints in descending order
(3) This table is then filtered for each borough and a 'limit' operation is applied on it to obtain the top 3 rows
(4) This leads us to the required view that contains the Top 3 Complaint Types for the filtered borough along with the count of complaints received for each type

Average Response Time for 311 Complaint Calls in each Borough:

(1) The Created Date and Closed Date attributes are used to find the response time for each complaint
(2) Response Time is computed to be the difference between Closed Date and Created Date
(3) The dataset is altered to retain the Closed Date, Create Date, Response Time and Borough columns
(4) Aggregation is done by Borough and AVG() is used to calculate the avg response time

Most frequent type of complaint in each Borough by year:

(1) The attributes 'Complaint Type', 'Borough', 'Year' (derived from 'Created Date') are retained from the base database
(2) A temp. table is created, aggregated by Complaint Type, Borough and Year to get the Count of Complaints
(3) The resulting relation provides information about the number of complaints received for each Complaint Type in a Year for each borough
(4) A window function is then applied on the temporary view created in the previous steps where the table is partitioned by Complaint Type for each Year and ordered in descending order by the count of complaints
(5) The table obtained in step 3 is further filtered to include only the highest occurring type of complaint for each borough by year using a 'WHERE' condition

6 CONCLUSIONS

In Task 1, using various techniques elicited above, we were able to achieve a certain level of accuracy. This accuracy can be further improved upon by cleaning the data to a much larger extent. This means, tightening the constraints and allowing only certain attributes to be allowed in those Columns.
The Goal of Task 2 was to perform a Semantic Check on the Columns in a dataset. The Semantics here could include name, phone number, address, etc. Various techniques were used to filter through the data such as Dictionaries, edit-distance, Levenshtein Distance, fuzzwuzzy.partialratio, fuzzywuzzy.ratio, API Calls, SoundEx, NYSIIS and many more. Here, like in Task 1, a certain accuracy was obtained but this too can be further improved upon. Improvement can be done by using stricter conditions which classifies a Column with a semantic type. A specific issue faced was that address columns were being matched with the semantic type Name. Upon further investigation, it was found that most street names in the dataset had names that corresponded to human names and therefore would be profiled as a name. This, although isn't wrong, it is also not right. Therefore this could be improved on in future work.
Task 3 involved analysis of 311 data. This dataset consisted of 22 Million rows and 40 columns. This required an hour to simply convert the TSV to a computer readable format. Operating on this dataset required a lot of prior Data Cleaning. For a dataset which had data from 2010 to 2019, it is almost guaranteed that there will be quite a bit of corrupt data present and therefore it is essential to perform data cleansing. All the rows and the columns which has a majority of wrong data would be dropped. This is to ensure that no false or erroneous conclusions are made. After cleaning the data, analysis using SparkSql was done and the necessary conclusions were formed. Average Response time per borough and highest number of types of complaints were found. This data can be very useful to the right people. For example, with this data, if there was a increase in average response time in a particular borough, then the city government can assign extra resources to that area to ensure faster response time.

7 AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: conception and design: WY, ML, SL; output collection: WY, ML; analysis: WY, ML, SL; interpretation of results: WY, ML, SL; draft manuscript preparation: WY, ML, SL

8 REFERENCES

(1) J. Leskovec, A. Rajaraman, and J. D. Ullman. Mining of Massive Datasets, 2nd Ed. Cambridge University Press, 2014.
(2) O. Lehmberg and C. Bizer. Stitching web tables for improving matching quality. PVLDB, 10(11):1502–1513, 2017.
(3) Imane El Alaoui, Youssef Gahi, Rochdi Messoussi, Youness Chaabi, Alexis Todoskoff, and Abdessamad Kobi. 2018. A novel adaptable approach for sentiment analysis on big social data. Journal of Big Data 5, 1 (2018)
(4) Mayank Bawa, Tyson Condie, and Prasanna Ganesan. 2005. LSH forest: self-tuning indexes for similarity search. Proceedings of the 14th international conference on World Wide Web - WWW 05 (2005).
(5) E. Rahm. Towards large-scale schema and ontology matching. Schema Matching and Mapping, pages 3–27. 2011
(6) Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, and Rajeev Motwani. 2003. Robust and efficient fuzzy match for online data cleaning. Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD 03 (2003).
(7) Michael Connor, Cynthia Fisher, and Dan Roth. 2012. Starting from Scratch in Semantic Role Labeling: Early Indirect Supervision. Cognitive Aspects of Computational Language Acquisition Theory and Applications of Natural Language Processing (2012), 257–296.
(8) Tamraparni Dasu, Theodore Johnson, S. Muthukrishnan, and Vladislav Shkapenyuk. 2002. Mining database structure; or, how to build a data quality browser. Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD 02 (2002).
(9) Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. 2013. Crawling deep web entity pages. Proceedings of the sixth ACM international conference on Web search and data mining - WSDM 13 (2013).
(10) Jaewoo Kang and Jeffrey F. Naughton. 2003. On schema matching with opaque column names and data values. Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD 03 (2003).
(11) Ilyas, A., Trindade, J. M. F. D., Fernandez, R. C., Madden, S. (2018). Extracting Syntactical Patterns from Databases. 2018 IEEE 34th International Conference on Data Engineering (ICDE).
(12) Zhang, M., Hadjieleftheriou, M., Ooi, B. C., Procopiuc, C. M., Srivastava, D. (2011). Automatic discovery of attributes in relational databases. Proceedings of the 2011 International Conference on Management of Data - SIGMOD 11
(13) Ramaswamy, S., Rastogi, R., Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. ACM SIGMOD Record, 29(2), 427–438
(14) Hua, M., Pei, J. (2007). Cleaning disguised missing data. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 07