Big Data Final Project - Data Cleaning at Scale

Report By: In-Memory Aggregators

SUYASH SONIMINDE, SYS8910, SYED AHMAD TAQI, ST4324, and VIHA GUPTA, VG2237

In this paper we present the use of data profiling and cleaning techniques on 14 NYC datasets. We propose generalized improvements to these techniques to enable scaling across a greater number of datasets. Both approaches are quantitatively compared in terms of efficiency. A link to our GitHub can be found below.

https://github.com/alekzanderx1/NYC-Open-Data-Profiling-and-Cleaning

Additional Key Words and Phrases: datasets, profiling, cleaning

ACM Reference Format:

Suyash Soniminde, sys8910, Syed Ahmad Taqi, st4324, and Viha Gupta, vg2237. 2021. Big Data Final Project - Data Cleaning at Scale: Report By: In-Memory Aggregators. 1, 1 (December 2021), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Just like the origins of trade and commerce in human civilization led to the formation of early algebra, the explosive growth of data in the 21st century has seen the formation of the concept of big data. According to the Oxford English Dictionary, data implies facts or information, especially when examined and used to find out things or to make decisions.[1] In the traditional mode, data was managed using the relational model with the focus on the basic 'ACID' properties of data- atomicity, consistency, isolation and durability. SQL(Structured Query Language) used to access the RDBMS(Relational Database Management System) was used extensively to manage the data. But the traditional mode suffered many disadvantages like lack of ability to work with unstructured data because of the static nature of the schema, no ability to do real time analysis and not being scalable enough to utilize the existing computational resources properly.

Big Data is the collection of theory, tools and processes we use to understand, manage, analyze and utilize the vast datasets and data streams that we encounter on a daily basis. Big Data solves the unstructured data problem by storing the data in the raw form only and applying dynamic schema while trying to access it. It enables real-time analytics. It is based on distributed architecture so it provides excellent scaling and great computational and economic benefits. As expressed in their seminal paper in 2004, to handle the large amount of data at Google, Jeffrey Dean and Sanjay Ghemawat came up with an abstraction to deal with parallelization complexity called MapReduce - one of the key paradigms of Big Data.[2]

Authors' address: Suyash Soniminde, sys8910; Syed Ahmad Taqi, st4324; Viha Gupta, vg2237.

Manuscript submitted to ACM

2 PROBLEM FORMULATION

Organizational Data is an extremely important asset for any organization as it heavily influences business decisions. A huge data deluge makes it difficult to manage the data. It then also substantially increases the risk of bad quality data which could cost any organization a substantial amount. Thus to reduce this risk it is very important to profile and clean the data to ensure it is of good quality. Data profiling helps us in exploring and getting a preliminary understanding of the data. Data Cleaning is the process of detecting and repairing corrupt or inaccurate records from a dataset in order to improve the quality of data.[3]

The first part of our problem is to explore the citywide payroll data and understand the data quality issues of the dataset. We then need to use the insights we gained to clean the dataset so that it correctly represents the facts and statistics. This updated dataset along with the understanding gained enable further accurate analysis and processing. For example if data quality issues had been properly identified and corrected, the 1999 Mars Meteorology mission would not have been muffed.[4]

In the second part we improve the strategies used in task 1 and scale them to handle a large number of files. We need to update the strategies used in task 1 and generalize them so that they scale well and can be used to handle data quality issues for a large number of datasets. By doing this we are creating a template that we can then reuse to avoid repetitive and tedious work. This also ensures a higher degree of certainty of data quality by nature of covering more edge cases.

3 RELATED WORK

3.1 Data Profiling Survey by Zaiwasch et al

The paper on Profiling by Ziawasch Abedjan, Lukasz Golab, Felix Naumann[14] gives us an idea about the different classifications of the data profiling task.

A. Single Column For a single column we could profile it using -

a1) Cardinalities : We can calculate the number of rows, the number of distinct values , number or percentage of null values, and uniqueness. Uniqueness is defined as the number of distinct values divided by the number of rows. We can also calculate the maximum, minimum, median and average values for the column.

a2) Patterns, Data Types and Domain : Patterns the data forms can be ascertained like deriving the regular expression for phone number data. Basic data types for the column can be derived like numeric, alphabetic, alphanumeric, date, time. Specific data types, such as varchar, timestamp and the maximum number of digits and decimals in numeric values can also be derived. Semantic domains can be identified like credit card, name ,phone number etc.

a3) Value Distributions : The cardinalities of a group of values like frequency histograms can be compiled. They can be equi-depth or equi-width. Equi-depth histogram can be used to give us the distribution of data in four quartiles. We can also calculate the distribution of first digits in the numeric data and check if that matches the one given by Benford's Law. This can be used to uncover fraud.

B. Multiple Columns For multiple columns we could profile it using -

b1) Correlations and association rules: Correlations and associations between columns can be identified . This can be pretty useful for recommender systems in e-commerce applications. For example, once we find that many people who buy bread also buy butter, we can recommend butter to anyone who buys bread.

b2) Clustering and outlier detection: We can use clustering to classify different homogeneous groups of records. This also identifies outliers which we can clean to improve the quality of the data.

b3) Summaries and sketches: We can also describe data using summaries and sketches . Data values can be sampled or hashed to smaller domains to achieve this. This can be very useful in answering approximate queries.

3.2 Data Cleaning example using OpenClean

Substantial work has been done to bring the different data cleaning tools under a single environment in the OpenClean Python library [5]. Openclean can be used effectively for the Data Profiling and Data Cleaning tasks. Here we describe the approach by which we can use OpenClean for the standardization of street names.

Goal : Finding groups of different street names that might be alternative representations of the same street.

Approach :

a1) Download the dataset.

a2) Use the streaming function to load part of the dataset into memory.

a3) Get a distinct set of street names from the column which contains street names in the dataset.

a4) Cluster street names using key collision clustering by choosing appropriate parameters. Use multi-threading while generating the value keys to take advantage of parallelization.

a5) The clusters can then be displayed to identify the groups of street names that might be alternative representations of the same street name.

Following is a snippet of the openclean code :

<>

```
# streets contains all the distinct street values
# minsize is the minimum number of distinct values that each cluster in the returned
# result has to have.
# threads is the number of parallel threads to use for key generation.

from openclean.cluster.key import key_collision

clusters = key_collision(values=streets, minsize=minsize, threads=threads)
```

<>

Fig. 1. Code Snippet

4 METHODS, ARCHITECTURE AND DESIGN

4.1 Initial Approach

For the first part of our project, we were tasked with profiling and cleaning the Citywide Payroll Data (Fiscal Year) provided by NYC OpenData[6]. To undertake this task, we decided to use the OpenClean[5] python library. By profiling our dataset we were able to get insights and understand our data. We found issues with null values, special characters, extreme values, and inconsistent letter casing. While architecting our solutions to these issues, however, our approach was very case specific.

For example, the column 'Agency Start Date' had nulls, and extreme outlier dates. To tackle nulls, we counted the number of nulls as a percentage of the total number of rows. Because the value was lower than our arbitrary threshold,

we decided to drop these rows entirely. Another issue we faced was with the date ranges. The values ranged from 1901-01-01 to 9999-12-31 (in yyyy-mm-dd). In the context of payroll data, we knew that an employee's retirement age in NYC is 62 years. If we assume that the minimum working age is 18, then a person can be working for at most 44 years. Since our dataset includes fiscal data only starting from 2014, then we dont expect to have data from before 1970 (2014 - 44). We added an arbitrary buffer of 5 years and thus we dropped rows that had 'Agency Start Date' older than 1965 (1970 - 5). We did a similar analysis for the other extremity. As the field inherently denotes a date that must be in the past, we also dropped any records that had future dates. We defined a future date as anything after June 30 2021 because the dataset uses this date to denote the end of the fiscal year.

Another column that had interesting data quality issues was that of 'Work Location Borough'. Although the dataset is meant to be representative of NYC's five boroughs, this field showed 23 distinct values which included the five boroughs but also blanks, 'Other' and places like Washington DC which are far from NYC. To tackle the problem of inconsistent casing, we first converted all values to uppercase. We replaced the blanks with the keyword 'UNSPECIFIED'. Because places outside NYC are out of scope for our dataset, we dropped them. Before blindly dropping all the data that wasn't in the set BROOKLYN, MANHATTAN, QUEENS, BRONX, STATEN ISLAND we noticed something peculiar. 'Staten Island' was missing completely from our dataset and 'Richmond' had exceptionally large frequency. After a quick google search we found that 'Richmond' was the old name for 'Staten Island' [7]. In order to keep the data future proof, we renamed all instances of 'Richmond' to 'Staten Island'. This was if and when 2022 fiscal year data is added with the new name, all old data will remain consistent.

4.2 Checking for Scalability and Improvements

For the second part of our project, we improved our existing techniques to make them more scalable and generalized for a larger number of datasets. To find potential areas of improvement, we looked for limitations in our previous approach.

Problems encountered include more number of special characters, a varied range of date formats and inconsistent letter casing for titles.

4.3 Architecture

We are using Apache Spark over HDFS on the NYU Peel cluster for this part. The methodology used is as follows:

- Two PySpark scripts are provided which clean a dataset at a time. These are as follows:
  - First script takes a sample of the dataset by using 95 percent confidence level and confidence interval of 8. The sample is then cleaned and both the original and cleaned sample is saved as csv on HDFS.
  - Based on the analysis done of the result of the first script, the second script has some improvements and additional checks to clean the dataset. This can either run on the entire dataset or a sample as mentioned above.
- Three Shell scripts are provided which run above mentioned PySpark scripts for each of the identified datasets. These perform the following tasks:
  - Clean any output from previous runs so that the execution doesn't fail.
  - Call spark-submit using the above mentioned scripts on every dataset in order.
  - For spark jobs which run on sample dataset, merge the output and provide them as csv for further analysis.

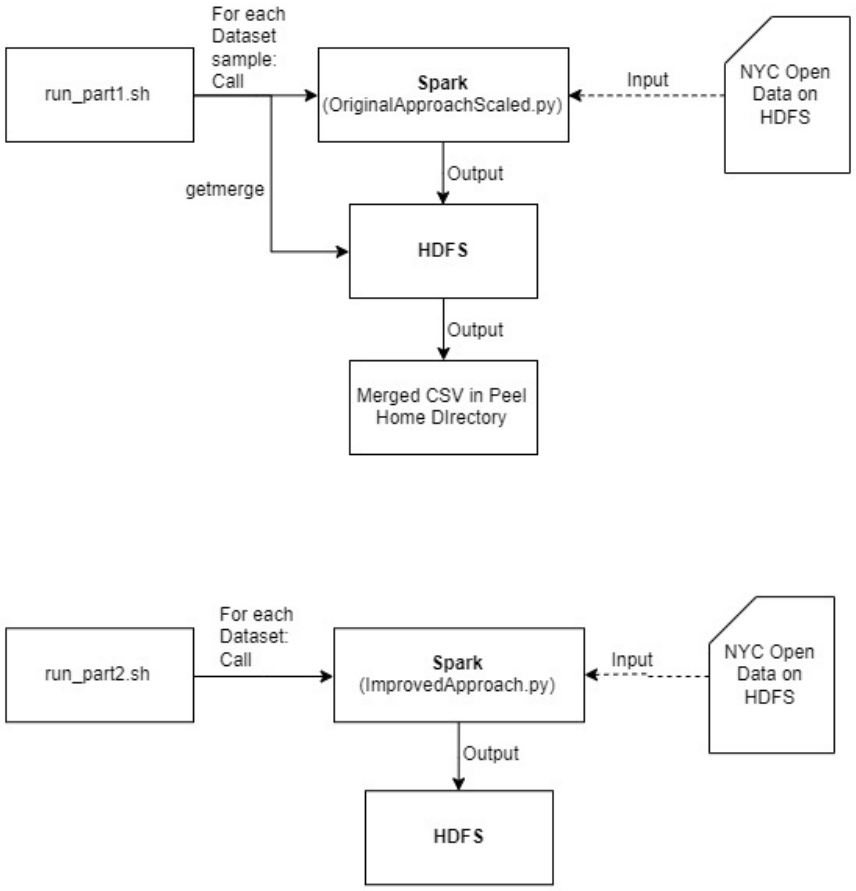

Fig. 2. Architecture

5 RESULTS

5.1 Efficiency Metrics

Where,

P = cleaned

N = not cleaned

TP = was relevant and successfully got cleaned

FP = was not relevant but got cleaned

TN = was not relevant and didn't get cleaned

FN = was relevant but didn't get cleaned

Confusion Matrix		Actual Values	
		Positive	Negative
Predicted Values	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

Precision : TP / (TP + FP)
Recall : TP / (TP + FN)

Fig. 3. Confusion Matrix

5.2 Sample Size

For each of the 14 datasets used, analysis of precision and recall was done using a sample size. Sample sizes were calculated on the basis of a 95 percent confidence level and 8 percent confidence interval.[10]

Dataset ID	Total Num of Rows	Sample Size
kpav-sd4t	2915	143
bdjm-n7q4	825691	150
p937-wjvj	2018180	150
pqg4-dm6b	1147	133
bquu-z2ht	665	123
nzjr-3966	3023	143
4d7f-74pe	104316	150
c5up-ki6j	3406	144
dsg6-ifza	29953	149
6khm-nrue	155	76
vz8c-29aj	14	13
ay9k-vznm	197	85
qdq3-9eqn	2097151	150
fudw-fgrp	74881	150

Fig. 4. Sample Size per Dataset

5.3 Initial Approach Results

For the analysis of results for the previous approach we demonstrate the precision and recall for a subset of three datasets, namely, NYC Jobs, Forestry Work Orders and NYC Civil Service Titles.

For Dataset "NYC Jobs" id: kpav-sd4t

NYC Jobs (Posting Date, Posting Updated, Process Date)		Actual Values	
		Positive	Negative
Predicted Values	Positive	150	0
	Negative	0	-

Precision : 1
Recall : 1

Fig. 5. Confusion Matrix for Posting Date in NYC Jobs

Our analysis of 'Posting Date', 'Posting Update' and 'Process Date' showed that all relevant records in the sample were correctly cleaned and that the format was properly changed to yyyy-mm-dd. The Precision and recall are both 1.0.

NYC Jobs (Post Until)		Actual Values	
		Positive	Negative
Predicted Values	Positive	0	0
	Negative	59	-

Precision : N/A
Recall : 0

Fig. 6. Confusion Matrix for Post Until in NYC Jobs

However, for the date column 'Post Until', 59 rows in the sample set have an unconventional date format like dd-Mon-yyyy that fail to convert to the form yyyy-mm-dd. As no rows were cleaned, the precision is undefined but recall is 0.

There are two title columns in this dataset 'Business Title' and 'Civil Service Title'. For 'Business Title', it so happened that all the rows in the sample set were clean to begin with. Thus, the precision and recall are both 0. For the 'Civil Service Title', we noticed 7 rows with dirty data of which 1 was correctly cleaned but 6 were missed. These 6 had a new kind of trailing special character "(" that we did not consider. Thus the precision is 1.0 and recall is 0.14. We will be adding this special character for our improvement script and then compare this same column as a benchmark.

The data set also contained two numeric type columns 'Salary Range From' and 'Salary Range To'. For both of these columns, we did not find any out of range values like negatives that need cleaning and so precision and recall were both 0.

For Dataset "Forestry Work Orders" id: bdjm-n7q4

NYC Jobs (Business Title)		Actual Values	
		Positive	Negative
Predicted Values	Positive	0	0
	Negative	0	-

Precision : 0
Recall : 0

Fig. 7. Confusion Matrix for Titles in NYC Jobs

NYC Jobs (Civil Service Title)		Actual Values	
		Positive	Negative
Predicted Values	Positive	1	0
	Negative	6	-

Precision : 1
Recall : 1/7

Fig. 8. Confusion Matrix for Titles in NYC Jobs

NYC Jobs (Salary Range From, Salary range to)		Actual Values	
		Positive	Negative
Predicted Values	Positive	0	0
	Negative	0	-

Precision : 0
Recall : 0

Fig. 9. Confusion Matrix for Salary Ranges in NYC Jobs

This dataset had a column for 'Borough'. All 157 rows were successfully cleaned and standardized to upper case. We achieved a precision and recall of 1.0.

We also performed our analysis on a few date columns, namely 'SanitationAssignedDate', 'SanitationRemovalDate', 'ClosedDate', and 'CreatedDate'. 'SanitationAssignedDate' had two rows with relevant columns and neither of them were cleaned to the proper format resulting in a precision and recall both of 0. 'SanitationRemovalDate' also had the same results of precision and recall but there happened to be no rows to clean. All rows were null.

The results for 'ClosedDate' and 'CreatedDate were interesting. Both columns had the same format of dd/mm/yyyy HH:MM:SS AM in the original dataset but only 'CreatedDate' was successfully converted to yyyy-mm-dd. It is suspected that this may be due to the presence of some null values in the 'ClosedDate' column. Clearly, this is a limitation of our technique. This is reflected by the precision and recall of 'ClosedDate' which are undefined and zero respectively.

Forestry Order (Borough)		Actual Values	
		Positive	Negative
Predicted Values	Positive	157	0
	Negative	0	-

Precision : 1
Recall : 1

Fig. 10. Confusion Matrix for Borough in Forestry Work Order

Forestry Order (Sanitation Assigned Date)		Actual Values	
		Positive	Negative
Predicted Values	Positive	0	0
	Negative	2	-

Precision : N/A
Recall : 0

Fig. 11. Confusion Matrix for Sanitation Assigned in Forestry Work Order

Forestry Order (Sanitation Removal Date)		Actual Values	
		Positive	Negative
Predicted Values	Positive	0	0
	Negative	0	-

Precision : 0
Recall : 0

Fig. 12. Confusion Matrix for Sanitation Removal in Forestry Work Order

For Dataset "NYC Civil Service Titles" id: nzjr-3966

For our final dataset analysis we examined the column of 'Title Description'. Of the ten relevant rows we found, four were infact cleaned by our technique while six were missed. These missed rows had special characters of "(" and "". We will be incorporating these in the next iteration. Although we achieved a precision of 1.0, the recall is only 0.4.

Finally, we have a look at the numeric columns such as 'MinimumSalaryRate', 'MaximumSalaryRate' and 'StdHours'. However, as none of the sample sets had negative values, no records were cleaned and the precision and recall are 0 for all.

Forestry Order (Closed Date)		Actual Values	
		Positive	Negative
Predicted Values	Positive	0	0
	Negative	135	-

Precision : N/A
Recall : 0

Fig. 13. Confusion Matrix for Closed Date in Forestry Work Order

Forestry Order (Created Date)		Actual Values	
		Positive	Negative
Predicted Values	Positive	157	0
	Negative	0	-

Precision : 1
Recall : 1

Fig. 14. Confusion Matrix for Created Date in Forestry Work Order

Civil Service Titles (Title Description)		Actual Values	
		Positive	Negative
Predicted Values	Positive	4	0
	Negative	6	-

Precision : 1
Recall : 0.4

Fig. 15. Confusion Matrix for Title Description in Civil Service Titles

5.4 Enhanced Approach Results

After applying our enhanced techniques to the NYC Jobs dataset, we observed that the previous 6 False Negatives (FN) of 'Civil Service Title' have been converted to True Positives (TP). This resulted in an increase of precision value to 1.0 from 0.14. The recall remained the same.

Similarly, the 'Title Description' column of the Civil Service Titles Dataset resulted in all 10 True Positives thereby increasing the precision from 0.4 to 1.0.

Civil Service Titles (Minimum Salary Rate, Maximum Salary Rate, Std Hours)		Actual Values	
		Positive	Negative
Predicted Values	Positive	0	0
	Negative	0	-

Precision : 0
Recall : 0

Fig. 16. Confusion Matrix for Salary Rates in Civil Service Titles

NYC Jobs (Civil Service Title)		Actual Values	
		Positive	Negative
Predicted Values	Positive	7	0
	Negative	0	-

Precision : 1
Recall : 1

Fig. 17. Enhanced Confusion Matrix for Title in NYC Jobs

Civil Service Titles (Title Description)		Actual Values	
		Positive	Negative
Predicted Values	Positive	10	0
	Negative	0	-

Precision : 1
Recall : 1

Fig. 18. Enhanced Confusion Matrix for Title Description in Civil Service Titles

5.5 Reference Data Repository

The dataset used for the reference data is "NYC Women's Resource Network Database". In this, the 'Organization Name' column which contains the list of NGO names was found to be a suitable candidate for the reference dataset. The approach used:

- Wrote a pyspark script
- Cleaned the column using previously identified techniques
- Some rows were found to contain sentences. These were handled by implementing a character limit of 61. This threshold was determined by manually inspecting the data
- Export the output of the pyspark script into a CSV file only for the relevant column

- This data is now ready to be used as a reference

6 FUTURE WORK

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

- 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-
  - 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.
  - 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.
- 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 the following-
  - 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.
- 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:
  - For Borough names also consider the case of Typos and fix them using similarity analysis.
  - 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.
  - 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.

ACKNOWLEDGMENTS

Thanks to professor Juliana Freire for teaching us and providing us with hands-on experience about the various algorithms and processes and tools needed to handle Big Data processing. And also we are grateful for her help in helping us provide the materials and understanding the knowledge required to perform this project.

REFERENCES

[1] Oxford Definition of Data https://www.oxfordlearnersdictionaries.com/us/definition/american_english/data

[2] Jeff and Sanjay's Mapreduce paper https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

[3] Data Cleaning definition https://en.wikipedia.org/wiki/Data_cleansing

[4] Mars Muffle https://www.wired.com/2010/11/1110mars-climate-observer-report/

[5] Openclean https://github.com/VIDA-NYU/openclean

[6] Citywide Payroll Dataset https://data.cityofnewyork.us/City-Government/Citywide-Payroll-Data-Fiscal-Year-/k397-673e

[7] Richmond Being Coextensive with Staten Island https://gothamist.com/arts-entertainment/staten-island-wasnt-officially-called-that-until-1975

[8] OpenClean Profiling Example https://github.com/VIDA-NYU/openclean-core/blob/master/examples/notebooks/nyc-restaurant-inspections/NYC%20Restaurant%20Inspections%20-%20Profiling.ipynb

[9] OpenClean Cleaning Example https://github.com/VIDA-NYU/openclean-core/blob/master/examples/notebooks/nyc-restaurant-inspections/NYC%20Restaurant%20Inspections%20-%20Cleaning.ipynb

[10] Sample Size Calculator https://www.surveysystem.com/sscalc.htm

[11] Auctus https://auctus.vida-nyu.org/

[12] PySpark Documentation https://spark.apache.org/docs/latest/api/python/

[13] Precision Recall Wiki https://en.wikipedia.org/wiki/Precision_and_recall

[14] Data Profiling Concepts https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/publications/2015/dataprofiling_main.pdf