Unsupervised Data Quality Monitoring: Three Approaches to Determine Batch Acceptability

Sheetal Laad, Micaela Flores, Ziv Schwartz, Kelly Sooch
New York University
{sl7054,mrf444,zs1349,kas1080}@nyu.edu

ABSTRACT

Since data engineering as a discipline largely requires extensive cleaning of raw datasets, there exists an increasing need for heightened data quality monitoring within engineering pipelines. In this paper, we present three methods of unsupervised data quality monitoring that determine, given a chronological series of datasets, whether a subsequent dataset in the series is of acceptable quality. We first establish the Baseline method to determine data quality acceptability, then create two new methods (which we name TFDV and Range) against which to compare the Baseline method. For all three implementations, we compare their accuracies on a model trained with varying batch sizes of training data. Our final results compare the methodologies and discuss the successes and difficulties of the different approaches.

1 INTRODUCTION

With the rise of big data, data files have become increasingly large in size, often times reaching petabytes in scale. For extremely large commercial companies, big data challenges include "anomaly detection, sampling accuracy, resource scheduling, and quantities of data too big for a single data center" [1]. Typically in data engineering pipelines, much of the effort is directed toward cleaning raw data from source databases prior to storing it in its final destination, a data warehouse. An essential facet of data cleaning is ensuring data quality in the data encountered. As data can be read in real-time via streaming or periodically in batches, it is usually recommended that it be loaded incrementally in batches. This is largely because data pipelines often leverage distributed systems where processing data in the wrong order could lead to data errors [2]. As such, for this project, our data will be read in as batches. Moreover, when working with large amounts of data, it becomes too difficult to manually check if each input data batch maintains its integrity. Thus, we provide a dynamic data auditing system that performs unsupervised data quality monitoring given a chronological set of clean and dirty data batches.

We employ various methods to define a dataset's acceptance criteria, including metadata analysis and data profiling (examining data from an existing source to collect statistics and other information). The data provided for this project consists of a Flights dataset and a Facebook posts dataset, which each have clean data batches along with corresponding dirty batches. The clean data represents data of acceptable quality, and the dirty data represents data that is not of acceptable quality. Our goal is to create a system that determines which data is acceptable and which is not acceptable based on the clean data files only. One of the challenges we face in this project is that the Flights dataset contains one dirty batch and one corresponding clean batch that are significantly different from the other dirty and clean batches in the dataset. As a result, there is not a clearly established divide between all of the clean and dirty data batches.

Our approach consists of a brief exploratory analysis of the clean and dirty datasets, followed by in-depth development of three data monitoring methodologies. All three methods consist of reading in training data batches, which are the clean data, and setting acceptable criteria based on this data. Then, if the subsequent test batch fits this criteria, the test data is classified as acceptable. Our final results demonstrate significant improvement upon the Baseline model, and we achieved the following goals.

- Achievement 1: Developed methodology to accurately classify batches of chronological data as acceptable or not acceptable.
- Achievement 2: Utilized multiple open source packages for creating and validating different data quality metric approaches.
- Achievement 3: Derived two ways of training models (using increasing and rolling batch sizes) before testing our classification methods on both datasets.

2 PROBLEM STATEMENT & APPROACH

2.1 Problem Statement

The problem we focus on answering is determining whether a subsequent batch of data has acceptable data quality. As there is no single characteristic to determine the acceptability criteria, we present here an exploration of various methods that can be used.

2.2 Approach

We consider several different approaches to establish the best methodology of correctly classifying acceptable versus

non-acceptable data. Depending on which method is used (Baseline, TFDV, or Range), the metrics we consider are the percentage of all nulls present in a single data batch, the expected values present in each column, and the uniqueness and completeness across columns. We approach the solution to the problem from both a training and criterion standpoint. Detailed descriptions of each method follow.

2.2.1 *Training Method.*

For training, we highlight two different training methodologies: Rolling and Increasing. The Rolling method allows for consistency in maintaining batch size and sees if other factors, such as seasonality and recency of specific data batches, provide more details. The Increasing method is useful as it sequentially adds more data to be trained on and allows the model to learn more information as each new batch is added.

2.2.2 *Criterion Method.*

Baseline: To establish a baseline model, we consider a rather naive approach of generalizing the prevalence of null values in an entire data batch. We would expect that dirty data should have a higher percentage of null values. We determined a threshold range for the percentage of nulls based on the training data, and if the amount of nulls in the test batch is outside of this range, we assign it as not acceptable. This methodology is established to create a simple and intuitive baseline model that classifies clean and dirty data files.

TFDV: In the next approach, we use TensorFlow Data Validation (TFDV) to determine whether each batch of data is acceptable or not. In general, a single TFDV pipeline consists of inferring the schema from an input data file and then checking if a subsequent data file follows the previously inferred schema. This is done by verifying that the new data file contains the column names of the input file and that the values in each column of the new data fall within the expected ranges or unique values of the inferred data. If this is not the case, then TFDV flags this new file as having anomalies. At this point, it is up to the user to manually check these anomalies and adjust the schema constraints accordingly. This can be done by adding foreign values to the schema or by relaxing the percentage of data values that must adhere to the schema. As discussed further in the sections below, we employ TFDV by reading in training data batches and updating the schema accordingly, and then comparing this to the schema of the subsequent test batch to determine its acceptability.

Range: In the final approach, we use training batches to create acceptable ranges for specific metrics in order to determine whether a subsequent test batch is acceptable. Our preliminary research suggests that completeness and uniqueness are strong indicators of data quality in a production pipeline. The completeness ratio is calculated separately for every field in a data batch. We define completeness as follows: count of records that have a missing value over the count of records that do not have a missing value. By this definition, we expect a dirty batch to have a higher ratio. Uniqueness is also calculated separately for every field in a batch by counting the number of distinct values in that field. For example, if the *Date* field contains only the values `Dec.04_2019` and `Dec_04_2019`, then that would be considered to have a distinct value count of 2. To get a sense of the difference between clean and dirty data batches in our datasets, we aggregate the mean values across all batches for each given column. This reveals a clear distinction between the clean and dirty data. Some column averages do not change, but overall, one can see a rather distinct difference between clean and dirty. Figure 3 and Figure 4 demonstrate the resulting completeness and uniqueness column averages for the Flights data.

3 IMPLEMENTATION

We experiment with two training methods, three criterion methods, and ten batch sizes over two different datasets.

3.1 Training Method

In the Rolling methodology, a predetermined number of batches is used for the training criterion. So, if the batch size is set to 3, then the first example would get trained on clean batches 1 to 3 and batch 4 will be predicted as acceptable or not. The second example would get trained on clean batches 2 to 4 and batch 5 will be predicted. In the Increasing methodology, the method trains on an increasing batch size. For example, with a starting batch size of 3, the first example would get trained on clean batches 1 to 3 and batch 4 will be predicted. For the second example, clean batches 1 to 4 will be used to train and batch 5 will be predicted.

3.2 Criterion Method

3.2.1 ***Baseline:*** For this method, we calculate the percentage of null values in each batch in the training data, and determine whether the subsequent test batch is acceptable or not based on these values. This percentage is a total value calculated on the entire batch, not separated by column. If the percentage of nulls in the test batch is not within the minimum and maximum percentage of nulls seen in the training batches, the batch is classified as not acceptable. Figure 5 in the Appendix shows the percentage of null values in each clean and dirty data batch. There is a clear distinction between clean and dirty data when looking at the percentage of null values across each batch.

3.2.2 ***TFDV:.*** In the TFDV pipeline, we begin by inferring the schema from the first training data file. Then, for every

subsequent training batch, our code reads in the data, infers its schema, and updates the existing schema with its information. When determining the schema constraints to relax, we first see whether the batch belongs to the Flights dataset or the Facebook dataset. In the Flights dataset, the columns `DepartureGate` and `ArrivalGate` were both reduced to 80% adherence to the trained schema and for the Facebook dataset, the columns `outlet` and `domain` were relaxed to 65% and 60% adherence, respectively. After this process, the number of anomalies is calculated again, using the same method as the first time. If the schema adjustment reduces the number of anomalies, then the test batch is considered acceptable.

3.2.3 ***Range:*** For this method, the model produces a range of acceptable completeness and uniqueness counts with which we can compare the same metrics of the test batches. Note that these intervals are created by respectively adding and subtracting two standard deviations from the mean of the training batch values. As can be seen in the Flights data in Figure 6, there is a large dip in distinct values for batch 14 for both the clean and the dirty versions. This makes it challenging to determine a range that would correctly classify this batch. We experiment with different scalars with which to multiply the standard deviation, but proceed with a factor of two as it creates the acceptability ranges with the best accuracy. Using this process, each field has its own acceptable range criterion. When a new batch is tested, if 80% of the fields are within the completeness range and 80% of the fields are within the uniqueness range, then the batch is deemed as acceptable. Refer to Figures 6 and 7 for how uniqueness and completeness for selected relevant fields in both datasets.

4 EVALUATION

4.1 Experimental Setup

For our experiments, we use the macOS operating system. We implement our models using Jupyter Notebooks (version 1.0.0) with Python 3.6. Likewise, the packages utilized in our code are Pandas (0.25.1), TensorFlow Data Validation (0.15.0), Matplotlib (3.1.2), Seaborn (0.9.0), NumPy (1.17.4), and Glob.

4.2 Datasets

We are provided with two datasets that are split into sequential batches: Facebook data (53 batches) and Flights data (31 batches). The Facebook data has 14 features (ex. page, week, number of likes, etc.) and the Flights data has 10 features (ex. flight number, scheduled departure, arrival gate, etc.). Each dataset contains clean and dirty batches, where the clean version is considered an acceptable batch while the dirty version is considered not acceptable. Differences between clean and dirty batches include higher missing value counts and more inconsistent value formats in the dirty batches.

4.3 Results

In order to compare the effectiveness of our methods, we evaluate each experiment on its accuracy. Accuracy is calculated as the number of correct classifications divided by the number of total batches tested. Figure 1 demonstrates an example of an experiment where 16 clean batches were incorrectly classified as unacceptable, while all the dirty batches were correctly classified as unacceptable. This resulted in an accuracy of around 72% for this experiment.

Figure 1: Results for Flights dataset using Rolling training method with batch size 2. See Figure 8 in the Appendix for enlarged plot.

Figure 2 shows the results of our experimentation with training methodology, criterion methodology, and batch size for both Flights and Facebook datasets. In regards to the training methodology, the accuracies for the Increasing method start off significantly higher than those in the Rolling method for all criterion methods. This result is expected, as the Increasing method continues to train on more data, while the Rolling method trains on the same amount of data in a shifting window. Generally for both Flights and Facebook datasets, the Increasing training method does not significantly improve accuracy as the starting batch size increases; however, the Rolling training method shows improvements from 50% to 100% accuracy.

For the criterion methods, the Range method outperforms the Baseline method in all scenarios. The TFDV method mostly outperforms the Baseline method, while at other times it underperforms the Baseline. Although TFDV outperforms Range with high batch sizes in the Flights Increasing method, generally the Range method was our best performing criterion method. Overall, as batch size increases, accuracy also increases until it appears to plateau. At this point, the accuracy does not improve significantly, meaning there exists a point after which no further training is needed. In classical machine learning terms, this point would be the optimal batch size needed to ensure satisfactory accuracy.

5 DISCUSSION

This project served as an exploration and evaluation of methods to validate the quality of predetermined clean and dirty

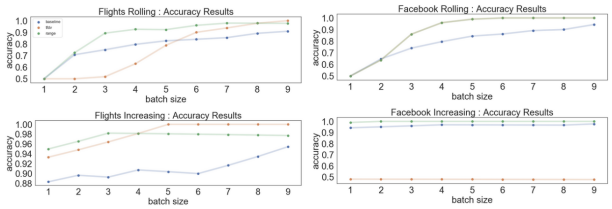

Figure 2: Accuracy results for all datasets using both training methods and all criterion methods. See Figure 9 in the Appendix for enlarged plot.

batches of data. Throughout our development and testing of this methodology, we achieved promising results, but also encountered some unexpected difficulties. Overall, both TFDV and Range methods significantly improved on the Baseline for all combinations except on the Facebook dataset with Increasing method, where TFDV did not outperform the Baseline. Furthermore, the methods that outperformed the Baseline also achieved objectively high accuracy for each batch size, often finishing in the upper 90% range.

The results of this project provide great insight into data quality monitoring in a production environment. While an increase in training batch size improves accuracy, the plateau (seen in Figure 2) suggests that there is an optimal batch size that should be used. Using additional batches for training when not required would unnecessarily increase runtime. Similarly, because the Increasing and Rolling training methods end up performing similarly, it would make more sense to use Rolling in a production environment. Again, this would allow for quicker runtimes, but more importantly it would give less weight to older information. This would be extremely helpful in production when a field starts getting recorded differently, and it would be able to capture temporal or seasonal changes in data.

Inherent differences in production batches, as we experienced with the Flights data, make data quality monitoring an especially difficult task. While our proposed methodologies dealt with this issue adequately, they are not robust to drastic changes between batches. Other difficulties arose due to the TFDV method having an extremely slow runtime. The Range and Baseline methods finished in approximately 30 minutes, whereas TFDV took over 24 hours for either Increasing or Rolling methods. This was likely due to the package itself not being designed for repeated schema adjustments at such large frequency as was used for this project. In general, it should be noted that the logic for determining data quality acceptability is the same across all methods except for TFDV. The criteria for a batch being labeled as acceptable or not using TFDV is different depending on which dataset is used. As mentioned in the Implementation section, we manually looked at the inferred schema to determine which column constraints to relax for each dataset. As a result, applying the TFDV method to any other dataset would require the same inspection process and would most likely be inefficient.

Next steps for this project include determining the quality of the data batches by looking more at the data values themselves rather than looking at the attributes of the data. This could include looking at minima and maxima that are expected for certain fields, or only allowing certain categorical values for other fields. While the TFDV method most closely resembles this idea, we emphasize that it is not the most optimal solution.

6 DETAILED CONTRIBUTIONS

Sheetal Laad Implementation: wrote and developed code for the Range criterion method; wrote and developed the framework for Training and Increasing training methodologies, accuracy calculation, and all visuals. Paper: wrote Range method sections, Results, part of Discussion; helped to edit paper.

Micaela Flores Implementation: wrote and developed all code for TFDV implementation, extensively ran analysis functions for testing TFDV methodology. Paper: wrote Abstract, Introduction, all TFDV sections in Approach and Implementation, Discussion; edited all sections, integrated figures into text and Appendix.

Ziv Schwartz Implementation: wrote and developed code for the exploratory analysis and methodology that helped with experiments regarding the Range methodology. Paper: wrote the Problem Statement, the training methodology and exploratory methods in the Approach and Implementation, and in the Range model. Edited all sections of the paper.

Kelly Sooch Implementation: wrote and developed code for Baseline method, helped experimenting with acceptable ranges in Range method. Paper: wrote Baseline sections, Experimental Setup, part of Introduction, and edited all sections.

REFERENCES

[1] Wiener & Bronson. "Facebook's Top Open Data Problems". Facebook Research, 22 Oct. 2014, https://research.fb.com/blog/2014/10/facebook-s-top-open-data-problems/

[2] Pathak. "ETL — Understanding It and Effectively Using It". Medium, 7 Jan. 2019, https://medium.com/hashmapinc/etl-understanding-it-and-effectively-using-it-f827a5b3e54d