# A Platform Solution of Data-Quality Improvement for Internet-of-Vehicle Services

Mingming Zhang
BDBC, School of Computer Science
Beihang University
Beijing, China
zhangmm@act.buaa.edu.cn

Tianyu Wo
BDBC, School of Computer Science
Beihang University
Beijing, China
woty@act.buaa.edu.cn

Tao Xie
Department of Computer Science
University of Illinois at Urbana-Champaign
IL, USA
taoxie@illinois.edu

***Abstract*—Interconnection and intelligence have become the latest trends of the new generation of vehicle and transportation technologies. Applications built upon platforms of cloud-centered vehicle networking, i.e., Internet-of-Vehicles (IoVs), have been increasingly developed and deployed to provide data-centric services (e.g., driving assistance). Because these services are often safety critical, assuring service dependability has become an important requirement. In this paper, we propose *DQI*, a platform-level solution of Data-Quality Improvement designed to assure service dependability for Internet-of-Vehicle services. As an example, *DQI* is deployed in CarStream, an industrial system of big data processing designed for chauffeured car services. Via CarStream, over 30,000 vehicles are organized in a virtual vehicle network by sharing vehicle-status data in a near real-time manner. Such data often have low-quality issues and compromise the dependability of data-centric services. *DQI* includes techniques of data-quality improvement, including detecting outliers, extracting frequent patterns, and interpolating sequences. *DQI* enhances the dependability of data-centric services in IoVs by addressing the common data-quality requirements at the platform level. Upper-level services can benefit from *DQI* for data-quality improvement and reduce the complexity of service logic. We evaluate *DQI* by using a three-year dataset of vehicles and real applications deployed in CarStream. The result shows that compared with existing approaches, *DQI* can effectively restore missing data and correct anomalies with more than 30.0% improvement in precision. By studying multiple real applications, we also show that this data-quality improvement can indeed enhance the dependability of IoV services.**

***Index Terms*—Dependability, Interpolation, Sequence Matching, Data Quality, Big Data, Internet-of-Vehicles**

## I. Introduction

The recent development of mobile computing technologies brought a revolutionary change to the automotive industry. Modern vehicles become more intelligent by embedding onboard computers and acquiring cloud intelligence through connecting to a cloud-centered IoV system [1]. In such a system, vehicles connected to the cloud upload vehicle statuses and request services from the servers through in-vehicle wireless modules or mobile phones. Meanwhile, the cloud collects, integrates, and analyzes data from the vehicles, and then provides macro-level situations of the transportation back to the vehicles [2]. By using these services, drivers can gather environmental information far beyond what the traditional approaches can provide, and have a better view of the road, other vehicles, and pedestrians. With such services, drivers may have more time to prejudge whether the current driving situation is unsafe and take actions accordingly. Being related to driving assistance, the services provided by IoV applications are often safety critical, and thus assuring the dependability of such services is very important.

In industrial IoV systems, the collected vehicle data usually suffer from low-quality issues, e.g., missing data, outlier data, and disorder. The causes of these issues can be multiple. Besides the sensor inaccuracy and malfunction (e.g., the connection loosening of data-collection devices), the accuracy of the driving data is also largely affected by the driving environment. For example, driving on a bumpy road may significantly change the reading of the gasoline sensor.

Such issues of low data quality may compromise the dependability of data-centric services provided by IoV systems from three aspects. First, the outlier and missing data may cause the services to generate wrong or inaccurate results. These faulty results directly affect the dependability of services and may even further cost human lives. Second, the applications running on IoV platforms need to deal with the data-quality issues in their processing logic. In this manner, the complexity of the processing logic could be increased and hence the applications become more vulnerable. Third, multiple applications may need to implement similar data-quality-related preprocessing functionalities independently such that developers need to maintain similar code multiple times. When a new data-quality issue comes up, each application has to be updated accordingly. Industrial IoV systems can be unreliable due to such problems and can put the services at risk. Facing such problems, the traditional solutions of service dependability assurance (such as transaction logging [3] and duplication deployment) may not be sufficient since the problems may largely come from the data perspective.

To address such issues, we propose *DQI*, a solution of data-quality improvement at the platform level for assuring the service dependability of CarStream [4]. CarStream is an IoV system with over 30,000 vehicles connecting and sharing vehicle-status data. *DQI* preprocesses the uploaded data in real-time, improving the data quality before the data are used by applications. To that end, *DQI detects* bad data (e.g., outlier, missing sequences) and *fixes* such bad data by interpolation with frequent patterns generated from the historical dataset.

In addition, *DQI* integrates a monitoring subsystem for data-processing topology. We evaluate *DQI* with a real-world dataset and apply *DQI* as part of the CarStream platform.

The paper makes the following main contributions:

- We identify dependability assurance of IoV services as addressing data-quality issues and propose a platform-level solution named *DQI*.
- We propose a real-time technique based on frequent patterns to improve the quality of IoV data.
- We evaluate the efficiency of *DQI* in a real-world dataset and an industrial IoV system.

## II. Dependability Characteristics of CarStream

As a data-centric IoV system, CarStream collects the data from vehicles through an on-board device named OBD connector, and provides services based on the analysis of the data. Services provided by CarStream are either safety critical or require high accuracy. This situation makes the dependability of CarStream even more important. The correctness and timeliness of services highly rely on the data quality.

However, the data that CarStream has collected from the vehicles are typically characterized as of low quality, making it difficult to build dependable services. In practice, the low-quality issues can be seen when the collected data are disordered, deviating from the actual value, insufficient in quantity, or delayed in time. Multiple factors can cause these issues. For example, the accuracy of the gasoline-level data is largely affected by the road condition and the driving style; when the road is bumpy or the driver drives aggressively, the sensors may produce jitter readings. Driving in regions with weak signals or poor contact of sensor connectors may cause the missing of data sequences. Such data missing also compromises the service dependability. In the dataset of CarStream, nearly 1.0% of data sequences contain missing parts; such proportion of missing may not be critical for the overall fleet, but still could significantly affect individual users. Figure 1 shows a statistical distribution of the length of the missing data subsequences. Another issue with the data is outliers. According to our empirical investigation, nearly 0.8% of the data can be identified as outliers. For example, the engine Round Per Minute (RPM) should be lower than 5000 in normal cases, but values larger than 8000 are observed; such high RPM values are impossible under normal road conditions and affect the accuracy of the applications that rely on these data.

Such issues pose challenges for CarStream as a dependable industrial system because the accuracy of its services can be compromised. For example, outlier data may increase the false alarms of detecting abnormal driving behaviors. The missing data can also cause applications not to be able to deliver results timely. For example, the vehicle anti-theft application, such as the electronic fence, may not generate timely alerts when some data are missing, and hence may cause property loss.

## III. Dependability Solutions in CarStream

Based on our analysis of the data characteristics and the dependability requirements of the applications in CarStream,

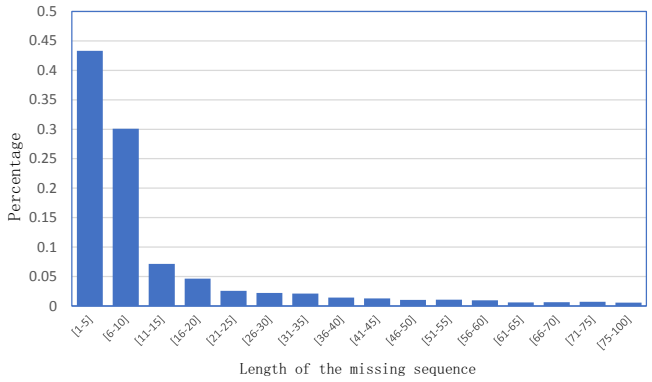


Fig. 1. Length distribution of missing subsequences in CarStream.

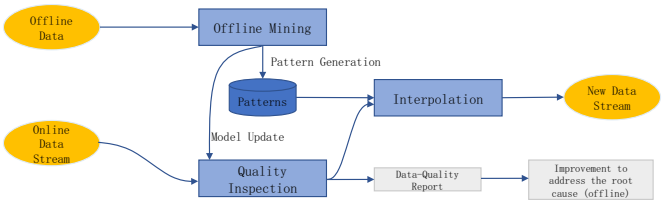


Fig. 2. Workflow of data-quality improvement

we identify dependability assurance of data-centric services as addressing data-quality issues and propose a solution of data-quality improvement to address such issues. We implement this solution as *DQI*, a subsystem that can significantly improve the dependability of CarStream. In this section, we discuss the design and implementation of *DQI*. We also include a lightweight monitoring module for data-processing flow in *DQI* to effectively achieve a robust platform.

### A. Data-Quality Improvement for Service Dependability

The data in CarStream are mainly about vehicle statuses. To provide dependable IoV services, we propose a workflow of data-quality improvement, as shown in Figure 2. Such workflow includes three parts: data-quality inspection, data-quality improvement, and pattern generation. Each part corresponds to an algorithm. The basic idea of this workflow is to detect and modify/interpolate the anomaly/missing data with frequent sequences generated from the historical dataset.

**Problem Statement.** Given a sequence $S =< d_1, d_2, d_3, \#, \#, ..., \#, \#, d_9, d_{10} >$ in which subsequence $S^* =< \#, \#, ..., \#, \# >$ represents a missing or anomaly subsequence that needs to be modified, the goal is to fill up $S^*$ with a subsequence that is closest to the original normal subsequence. Note that $i$ in $d_i$ represents the time stamp of the data.

**Quality Inspection.** This part detects missing subsequences and abnormal subsequences. By comparing the time stamps of the data in the sequence, the inspection algorithm detects the missing subsequences and inserts placeholders $\#$ into the sequence such that the placeholders can be replaced with data by our interpolation algorithm. The inspection algorithm also detects abnormal subsequences with the probabilistic algorithm of outlier detection. The detection is based on data distribution [5]. Specifically, we calculate the extreme value

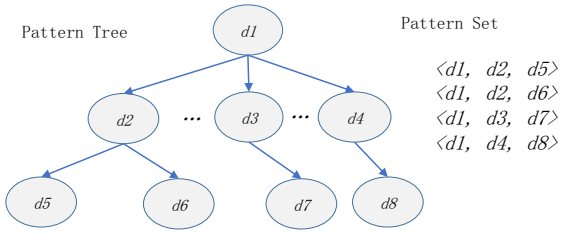


Fig. 3. An example of pattern index tree.

**Algorithm 1** Subsequence Interpolation Algorithm

**Input:** (1) $S_p$;(2) $S =< s_{pre}, s^*, s_{tail} >$.
**Output:** (1) $S =< s_{pre}, s^m, s_{tail} >$.

1: $MSE \leftarrow MaxNumber$
2: $S_p^* \leftarrow$ SearchTreeEntrance($S_p$,$s_{pre}$)
3: **for** each tree entrance $s$ in $S_p^*$ **do**
4: $s^p, s^m, s^t \leftarrow s$ //separate $s$ into subsequences
5: $MSE^* \leftarrow MSE(s^p, s_{pre}) + MSE(s^t, s_{tail})$
6: if ($MSE^* < MSE$)
7: $s^* \leftarrow s^m$ //find the pattern with minimum MSE
8: **end for**
9: output $S =< s_{pre}, s^m, s_{tail} >$

$P_{EV}$ of each data $d$ according to Eq. 1 and determine data anomaly by comparing $P_{EV}$ with the data distribution. We detect sequence disorder by comparing the time stamps of data with a delay-and-sort strategy locally in this module.

$$P_{EV}(d) = exp\{-exp(-\frac{d-\mu}{\sigma})\} \quad (1)$$

For streaming data, an anomaly could be either a point outlier or contextual anomaly [6]. Note that as a platform-level module, the inspection algorithm focuses only on point-outlier data instead of contextual anomalies. The reason is that a contextual anomaly may be caused by an abnormal driving behavior, and such anomaly should be detected at the application level.

**Pattern Generation.** The pattern-generation algorithm takes historical data as input and mining frequent subsequences, which are also called patterns. The generated frequent subsequences are used by the online interpolation to fill up the missing data or modify the detected outliers. This algorithm is based on *PrefixSpan* [7], a fast sequential pattern mining algorithm. The basic idea of *PrefixSpan* is that the sequence projection is conducted based on only frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix. Such algorithm generates a set of subsequences from the historical dataset, and this pattern set is further queried by the interpolation algorithm.

The performance of querying the pattern set is a key performance concern because once the pattern set is generated, it will be queried frequently. To accelerate the querying, we organize the patterns with an index-tree structure in memory. Figure 3 illustrates the structure of such index tree. In the tree, each node has one or more children nodes. Each pattern is represented as a branch of the tree. Because the patterns may share similar prefixes, part of the commonly shared prefixes can be represented with the same branch in the tree, thus reducing the memory cost.

**Interpolation.** Given a pattern set $S_p = \{s_1, s_2, ..., s_n\}$

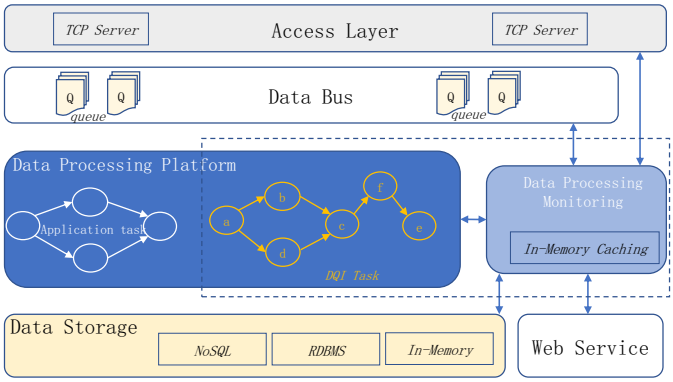


Fig. 4. An overview of CarStream

sequence $S =< s_{pre}, s^*, s_{tail} >$, where $s_{pre}$ and $s_{tail}$ are the prefix and tail subsequences, respectively, and $s^*$ is the missing or anomaly subsequence. The interpolation algorithm finds the most similar prefix match and tail match of $s$ from $S_p$, and then uses the corresponding part $s^m$ to interpolate $s^*$. We use mean square error (MSE) as the criterion of subsequence similarity. $MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$. The idea here is that, if the historical data are large enough, then each new sequence $s$ should have a corresponding pattern $\hat{s}$ in $S_p$ where $MSE(s, \hat{s}) < \theta$ such that the missing part of $s$ can be replaced with $\hat{s}$. $\theta$ is a threshold determined by the average similarity of the best matches in the dataset. The interpolation algorithm is shown in Algorithm 1.

The interpolation algorithm is invoked when there is any data missing or modifying placeholder found in the data sequence. For example, when # is observed from the sequence, the subsystem generates a pattern query with a buffered prefix subsequence $s_{pre}$ of length $prefixLength$, the placeholder subsequence $< \#, \#, ..., \# >$ (as $s^m$, whose length may vary), and a tail subsequence $s_{tail}$ of length $tailLength$. $S =< s_{pre}, s^*, s_{tail} >$ becomes the input of the algorithm, which then outputs $s^m$ as the interpolation/modification part of the sequence.

### *B. Implementation of Dependability-Assurance Infrastructure*

We implement the solution for data-quality improvement named *DQI* as part of the CarStream platform. Specifically, *DQI* is a data-processing task that runs on the data-processing platform. The *DQI* task contains two main parts: an offline batch-processing task that generates and updates the frequent pattern dataset, and an online stream-processing task that detects outliers and missing parts form the newly generated data, and interpolates the data based on frequent patterns. An overview of the CarStream system architecture is shown in Figure 4.

Applications of CarStream are also implemented as data-processing tasks running on the data-processing platform to provide data-centric services. In other words, *DQI* co-exists with application tasks. *DQI* takes the raw data from the queue in the data bus and generates an improved data queue back to the data bus. Such improved data queue can be further used by other application tasks.

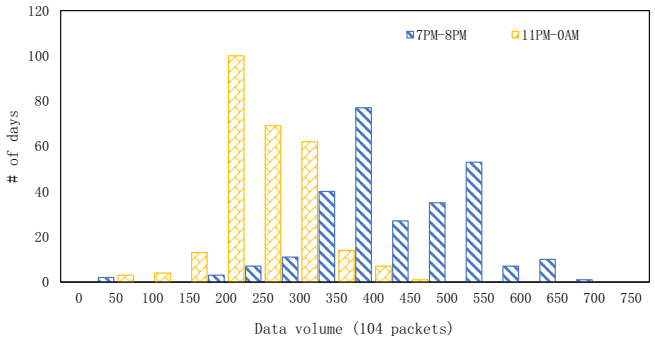


Fig. 5. Distribution of the data volume in two different hours (7PM-8PM and 11PM-0AM)

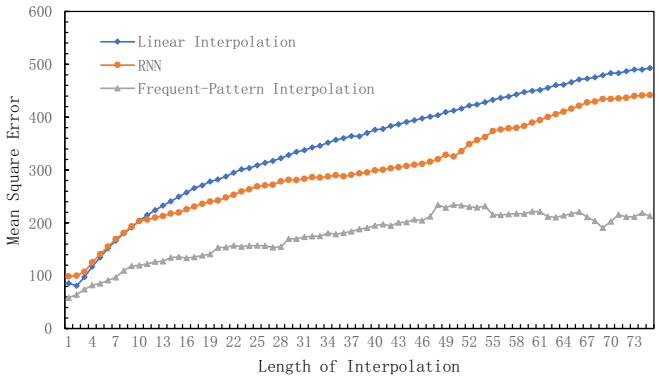


Fig. 6. The accuracy of the interpolation approaches when dealing with different lengths of data.

*DQI* addresses data-quality issues from a micro level. However, in CarStream, there are also some data-related issues that cannot be fixed automatically by *DQI*. For example, a network-connection failure may cause data uploading to stop, and a hardware malfunction can cause large-scale outlier data to be generated. Such issues may be fixed offline. Therefore, in addition to the data-quality improvement, we design a lightweight monitoring subsystem of data-processing topology for CarStream as part of dependability assurance. The reason for designing the topology-monitoring subsystem is that CarStream takes a popular stream-computing-based industrial architecture. In such a system, stream-processing tasks often have data dependency on each other: once a task encounters a problem, the problem would probably broadcast by causing other tasks to be in an idling status. In other words, the problem of one application may affect other applications. This situation makes it difficult for operators to locate the root cause of the problem within limited time. Therefore, real-time task-failure detection becomes significant for service-dependability assurance. The processing topology monitoring subsystem mainly monitors two aspects of each processing task: whether the task is still alive, and whether the processing task is working properly. To that end, the subsystem monitors the following attributes:

- **Heartbeat**. Heartbeat is a tuple *<moduleID, timestamp>* that is encapsulated by the processing task.
- **Input-Data Speed**. The input-data speed of a specific task reflects its input load and the output load of its precursor task.
- **Processing Delay**. The processing delay is the time difference between the data being generated by the client and being processed by the server.
- **Input/output Ratio**. The input/output ratio is helpful for inferring whether there is something wrong with the processing logic.

Each task is integrated with a lightweight monitoring module to maintain and update these parameters to a central monitoring server. By updating the *heartbeat*, it is easy to know the live status of each task. Meanwhile, the *Input-Data Speed*, *Processing Delay*, and *Input/output Ratio* are monitored to infer whether the task is processing the data to be in a normal status. For example, if a significant change in the *Input/output Ratio* of a task is observed, there might be a problem in the data or the processing logic of that task. Compared with logging each transaction of the data processing, the cost of monitoring such parameters can be neglected. Once a problem is detected, the subsystem sends an alert message to notify operators timely, so that further investigation can be done to prevent services from failures or recover from failures that occurred.

For the monitoring of data-processing topology, a key task is to determine the criteria for alerting, such as "what is an abnormal input-data speed?". In IoV scenarios, the characteristics of the data stream change over time. For example, the data speed in the peak hours could be up to 80 times larger than that in normal hours. Figure 5 lists the distribution of data speed in two different hours (7PM-8PM and 11PM-0AM) from January to October in 2016. Based on such fact, the topology monitoring subsystem takes a fine-grained criterion to construct the monitoring rules. Specifically, we divide a day into 24 hours. For each hour, we generate a rule that could determine whether the current monitored item is normal. Such rule generation is based on the statistics of the historical dataset.

## IV. Experimental Evaluation

### A. Experiments

**Data-Quality Improvement.** As shown in Figure 1, over 70.0% of the missing subsequences are within a length of 10 expected data intervals. Therefore, the interpolation algorithm has to at least deal with the missing length of 10-15 with high accuracy to make sure that a large proportion of abnormal data could be fixed. To test the prefix-based interpolation algorithm, we design a series of experiments with the interpolation length varying from 1 to 70. We compare the performance of the prefix-based algorithm with the linear interpolation algorithm. Linear interpolation is the most basic algorithm used to fix missing subsequences. Such algorithm simply takes the missing part as a straight line and interpolates the data by calculating the slope. For example, given $S =< d_i, s^*, d_j >$ and $length(s^*) = m$, then $s_i^* = i * (d_j - d_i)/m$.

We generate patterns (indexed in memory) from three months' historical data. For each interpolation length $l \in [1, 75]$, we use 50,000 sequences (without any issues) as test data. We manually split each sequence into three parts, $< s_{pre}, s^*, s_{tail} >$, where $s^*$ is considered being missing or the correct value for an anomalous value. Therefore, $s^*$ is used here as the ground truth of the interpolation algorithm. We use different approaches to generate $s^m$, an estimation of $s^*$. The approaches take $s_{pre}$ and $s_{tail}$ as input and output $s^*$. The $MSE$ of interpolation length $l$ is calculated by the $MSE$ of $s^*$ and $s^m$. $MSE = \frac{1}{n}\sum_{i=1}^{n}(MSE(s_i^*, s_i^m))$.

Besides our proposed approach, sequence prediction algorithms are also widely used in filling up missing values or modifying outlier data. Recent practices use Recurrent Neural Network (RNN) to predict a sequence. In such an RNN-based approach, a prefix sequence $s_{pre}$ is given to predict the succeeding sequence $s^*$. An RNN is trained as a sequence regressor. Here, we test the RNN-prediction-based interpolation based on *Tensorflow*. More specifically, we use *LSTM* (Long Short-Term Memory) [8] as the RNN model. The network contains two layers and each layer has 30 neurons. For each training dataset, the model iterates through 10,000 steps. In the experiments, for short sequences, the RNN-based approach gains similar accuracy with the approach of linear interpolation, i.e., the linear approach. Such result suggests that the accuracy of linear interpolation is not low. Because in IoV scenarios, a vehicle's status does not change dramatically within a short period, the data tend to maintain similarly and the linear approach can achieve a relatively low $MSE$. However, with the increase of interpolation length, the RNN-based approach achieves better result compared with the linear approach. Such result suggests that the RNN has successfully learned the patterns from the training data.

Figure 6 shows the result of three approaches in our experiments. The result indicates that the prefix-based approach outperforms both the basic linear approach and RNN-based approach, especially when dealing with long sequences. Such performance can be attributed to the pattern set, which represents most of the patterns of the sequences. The result also indicates that, with the increase of the interpolation length, the $MSE$ increases, indicating that the error increases. It is not difficult to understand such trend because the longer a missing sequence is, the more difficult it is to be predicted.

Figure 7 shows the distribution of $MSE$ for an arbitrarily selected interpolation length. The long-tail result complies with the expectation that most interpolations are accurate. Besides using the tree structure to index the patterns, we also use hashing to accelerate the search of the trees. The experimental result shows that a single thread can match a sequence within millisecond scale, and high throughput can be achieved with paralleled stream processing.

Compared with the linear approach, the RNN-based approach requires a huge cost to train the network, and the prefix-based approach requires a huge cost to both generate and maintain the patterns. Therefore, from this experimental result, we can learn that the linear approach is sufficient

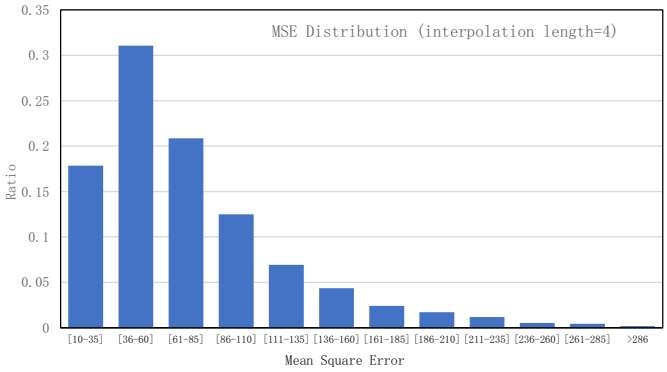


Fig. 7. The MSE distribution of prefix-based interpolation in a given sequence length.

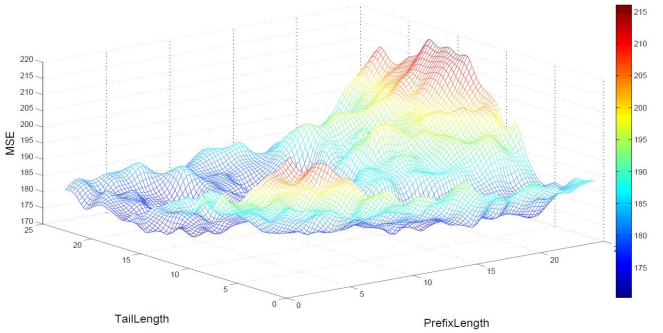


Fig. 8. The influence of prefix length and tail length on the accuracy of the prefix-based interpolation algorithm.

only for a scenario with short missing sequences and limited computing resource. If high accuracy of data interpolation is needed and the memory cost is not a concern, the prefix-based approach should be the choice. In CarStream, the importance of accuracy is much higher than the memory cost; therefore, we adopt the prefix-based approach.

For prefix-based interpolation, the algorithm first matches the prefix $s_{pre}$ and the tail $s_{tail}$ of the sequence with the pattern sequences in the pattern set, and finds the best match. Therefore, the length of $s_{pre}$ and $s_{tail}$ may have an influence on the accuracy of the result. Normally we would think that the longer the prefix and tail are, the more accurate the result is. However, the experimental result indicates differently, as shown in Figure 8. Such result may be explained by the periodic characteristics of driving (or the data sequence), the prefix and tail length influence the *entrance* of pattern matching, and hence may influence the result. This diagram can be used to help determine the algorithm parameters. A length between 10 to 15 is suggested for the tail and prefix on the real dataset.

**Topology Monitoring.** The out-of-band processing-topology monitoring subsystem detects events that could significantly affect the system dependability. Table I lists some events (in CarStream) that have been detected by the monitoring subsystem. Some of the events are originally caused by the software of the vehicle side. The software defects occurring on the client side may not be realized by developers on the cloud side in a timely fashion. Those events can be detected by the monitoring subsystem in the early

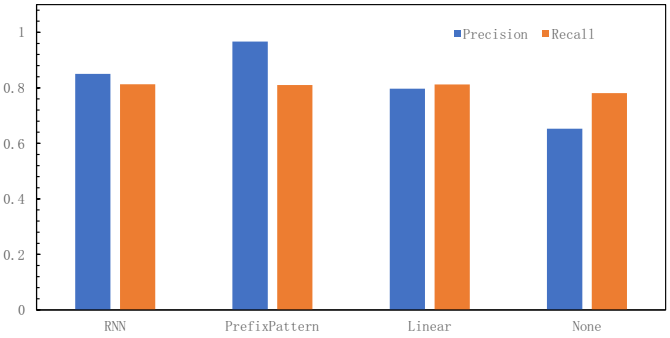


Fig. 9. The precision and recall of different approaches on detecting abnormal driving behaviors.

stage and alert developers to take actions accordingly, such as migrating servers or starting backup tasks.

### B. Case Study

We next discuss how service dependability is enhanced by DQI with two application scenarios.

**Detecting abnormal driving behaviors**. This application scenario detects contextual anomalies, e.g., rapid acceleration and nasty breaking, from the driving-data sequence.

In this application scenario, if the data contain outliers, such as speed jerks and RPM jerks, false alerts would be generated, even if the driver is driving in a normal status. Such false alerts compromise the accuracy of the services provided by the applications. By doing the data-quality improvement, the false alerts can be significantly reduced. An experimental result of detecting abnormal driving behaviors is shown in Figure 9. The recall here indicates the fraction of detected abnormal behaviors over the total number of abnormal behaviors. Without employing *DQI*, the precision of detecting abnormal driving behaviors is slightly over 65%. Such low accuracy can be attributed to the noise in the raw data. By using the three approaches of data-quality improvement, the precision has a significant improvement and the frequent-pattern-based approach achieves the best result. However, the recall is not improved much. The reason is that such data-quality improvement can only help lower false positives (*FP*) caused by the outlier data, but cannot change the false negatives (*FN*) of the detection. To improve *FN*, the anomaly-detection algorithm needs to be improved at the application level.

**Electronic-fence alert**. This application scenario allows users to draw a closed region on the map; once a vehicle leaves or enters this region, an alert will be sent to the users. Such application scenario is useful for anti-theft or fleet management.

In this application scenario, if a vehicle has been stolen and the vehicle is driving out of the pre-defined electronic fence, and the data sequence of this vehicle has stopped for some reason, the fence alert will not be triggered and no alert will be sent timely. The owner of the vehicle may lose the valuable time to trace the vehicle. However, if the system can detect such data loss and try to make a prediction (data interpolation) of the upcoming trajectory, then there is still a chance to trigger an alert. In our simulated test, even when the car suddenly stops uploading data, the service can still send alerts with the interpolated data.

## V. Related Work

Dependability of big-data systems has been an important focus for both industry and academia. Providing a dependable service requires tremendous efforts from multiple aspects such as improving data quality, monitoring, and robust design and deploying.

As a fundamental aspect, data-quality assessment has been well studied. Batini et al. [9] discuss the methodologies of data-quality assessment and improvement for information systems. To address the lacking of metrics in practice, Pipino et al. propose several principles that can help organizations develop usable data-quality metrics [10]. Sieve [11] is a framework for flexibly expressing quality-assessment algorithms of linked data as well as fusion algorithms. Data quality is highly related to the dependability of services. Cappiello et al. [12] propose an architecture for assessment and monitoring of data quality. Wang and Strong [13] develop a framework that captures data-quality aspects important to data consumers.

The techniques used to improve data quality in big-data systems usually include outlier detection [14] and missing-data prediction. Probability-distribution-based algorithms [15] are widely studied and adopted for outlier detection. Principle Component Analysis (PCA) is used in anomaly detection for years [5], [16], and Callegari et al. [17] propose an improved anomaly detection algorithm based on PCA. Clustering-based anomaly detection [18] is used on static datasets. Such algorithm groups similar data instances into clusters and those data that are not grouped into any clusters are considered as anomalous [19]. The assumption of this algorithm is that normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster. After evaluating data quality, the next step is to improve the data [20]. For sequential data, such as those in IoV, missing-data interpolation algorithms are widely adopted. Cai et al. [21] use RNN to predict data for time series. Xu et al. [22] design an approach that includes three algorithms (SUTR, QTR, and QTR+QR) to predict travel speed. These algorithms are used in different conditions to achieve high accuracy.

## VI. Conclusion

Assuring service dependability in an industrial IoV system can be challenging because data-quality issues may compromise the dependability of data-centric services, and addressing the data-quality issues in different applications independently may compromise the maintainability and robustness of the system. To tackle such issues, in this paper, we have identified data quality as an important influencing factor of the service dependability and have proposed *DQI*, a solution at the platform level for improving the dependability of data-centric services. In particular, *DQI* adopts a novel technique for data-quality improvement. Such technique uses outlier detection to

TABLE I
SOME REAL EVENTS THAT ARE DETECTED BY APPLICATION MONITORING IN CARSTREAM

| Symptom | Cause | Description |
|---|---|---|
| Input data speed suddenly increased | Misconfiguration | The sampling frequency of OBD connectors is misconfigured. The data-uploading speed is tripled, quickly exhausting the bandwidth and the buffer of message queues. |
| Input data speed suddenly dropped | Software upgrade | The batch upgrade of OBD software causes a sudden drop of input data speed within several minutes. |
| | Network or data connector failure | The input of a data-receiving node is observed dropping to 0 because its corresponding network link is down or the data-connector process is no longer working. |
| Input/output ratio significantly changed | Low data quality | Some vehicles stop uploading the corresponding parameter values. |
| | Software defect | Vehicles upload repeating data, which significantly change the input/output ratio of the data-filtering task, to the server because of a configuration error. |
| Processing delay largely increased | Database congestion | A defect in the connection-pool management of the database causes the congestion of data store and further congests its upstream processing tasks. |
| Processing task down | Server failure | Some processing tasks are down and stop updating heartbeat because the underlying server is down. |
| | Memory leak | The produced data of a processing task are accumulated because the downstream tasks cannot quickly consume the data. Such situation causes a memory leak and further kills the process. |

identify data anomalies, and uses frequent patterns to *fix* the *bad* or *missing* data. We have implemented *DQI* as a series of data pre-processing tasks providing the *improved* data source to applications for various computations. In addition to data-quality improvement, we have designed a processing-topology monitoring subsystem to detect data-processing malfunction and monitor the topology of data processing. *DQI* has been evaluated in CarStream with a real-world dataset, and the result shows an improvement of 30.0% for the precision of the services.

## ACKNOWLEDGMENT

This work is supported by grants from China Key Research and Development Program (2016YFB0100902), the National Natural Science Foundation of China (91118008, 61421003). Tao Xie's work was supported in part by National Science Foundation under grants no. CCF-1409423, CNS-1513939, CNS-1564274. We would like to thank UCAR Inc. for the collaboration and the data used in this paper.

## REFERENCES

[1] Y. Leng and L. Zhao, "Novel design of intelligent Internet-of-Vehicles management system based on cloud-computing and Internet-of-Things," in *Proceeding of the International Conference On Electronic and Mechanical Engineering and Information Technology (EMEIT)*, vol. 6. IEEE, 2011, pp. 3190–3193.

[2] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proceedings of the World Forum on Internet of Things (WF-IoT)*. IEEE, 2014, pp. 241–246.

[3] R. Ding, Q. Fu, J. G. Lou, Q. Lin, D. Zhang, and T. Xie, "Mining historical issue repositories to heal large-scale online service systems," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2014, pp. 311–322.

[4] M. Zhang, T. Wo, T. Xie, X. Lin, and Y. Liu, "CarStream: An industrial system of big data processing for Internet-of-Vehicles," in *Proceedings of the Very Large Data Bases Endowment (VLDB)*, vol. 10, no. 12, 2017, pp. 1766–1777.

[5] S. J. Roberts, "Novelty detection using extreme value statistics," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 146, no. 3, pp. 124–129, 1999.

[6] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Conditional anomaly detection," *Transactions on Knowledge and Data Engineering*, vol. 19, no. 5, pp. 631–645, 2007.

[7] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proceedings of the International Conference on Data Engineering (ICDE)*, 2001, pp. 215–224.

[8] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[9] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *Computing Surveys (CSUR)*, vol. 41, no. 3, p. 16, 2009.

[10] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data quality assessment," *Communications of the ACM*, vol. 45, no. 4, pp. 211–218, 2002.

[11] P. N. Mendes, H. Mühleisen, and C. Bizer, "Sieve: linked data quality assessment and fusion," in *Proceedings of the Joint Conference of the Extending Database Technology and the International Conference on Database Theory (EDBT/ICDT) Workshops*, 2012, pp. 116–123.

[12] C. Cappiello, C. Francalanci, and B. Pernici, "Data quality assessment from the user's perspective," in *Proceedings of the International Workshop on Information Quality in Information Systems*, 2004, pp. 68–73.

[13] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5–33, 1996.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[15] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

[16] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[17] C. Callegari, L. Gazzarrini, S. Giordano, M. Pagano, and T. Pepe, "Improving PCA-based anomaly detection by using multiple time scale analysis and kullback–leibler divergence," *International Journal of Communication Systems*, vol. 27, no. 10, pp. 1731–1751, 2014.

[18] Tan, Pang-Ning, Steinbach, Michael, Kumar, and Vipin, *Introduction to data mining*, 2016, vol. 22, no. 6.

[19] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the International Conference on Data Mining (ICDM)*, 2003, pp. 47–58.

[20] H. Müller, J.-C. Freytag, and U. Leser, "Improving data quality by source analysis," *Journal of Data and Information Quality*, vol. 2, no. 4, p. 15, 2012.

[21] X. Cai, N. Zhang, G. K. Venayagamoorthy, and D. C. Wunsch, "Time series prediction with recurrent neural networks trained by a hybrid pso–ea algorithm," *Neurocomputing*, vol. 70, no. 13, pp. 2342–2353, 2007.

[22] B. Xu and O. Wolfson, "Time-series prediction with applications to traffic and moving objects databases," in *Proceedings of the International Workshop on Data Engineering for Wireless and Mobile Access*, 2003, pp. 56–60.