MIS-IoT: Modular Intelligent Server Based Internet of Things Framework with Big Data and Machine Learning

Conference Paper · December 2018

DOI: 10.1109/BigData.2018.8622247

4 authors:

Aras Onal
TOBB University of Economics and Technology
3 PUBLICATIONS **40** CITATIONS
SEE PROFILE

Omer Berat Sezer
TOBB University of Economics and Technology
18 PUBLICATIONS **333** CITATIONS
SEE PROFILE

Murat Ozbayoglu
TOBB University of Economics and Technology
99 PUBLICATIONS **868** CITATIONS
SEE PROFILE

Erdogan Dogdu
Angelo State University
82 PUBLICATIONS **811** CITATIONS
SEE PROFILE

MIS-IoT: Modular Intelligent Server Based Internet of Things Framework with Big Data and Machine Learning

Aras Can Onal*, Omer Berat Sezer*, Ahmet Murat Ozbayoglu*, Erdogan Dogdu†
* Department of Computer Engineering
TOBB University of Economics and Technology
Ankara, Turkey 06560
Email: {arascanonal, oberatsezer, mozbayoglu}@etu.edu.tr
† Department of Computer Engineering
Cankaya University, Ankara, Turkey 06790
Georgia State University (adj.), Atlanta, GA, USA 30302
Email: {edogdu}@cankaya.edu.tr

***Abstract*—Internet of Things world is getting bigger everyday with new developments in all fronts. The new IoT world requires better handling of big data and better usage with more intelligence integrated in all phases. Here we present MIS-IoT (Modular Intelligent Server Based Internet of Things Framework with Big Data and Machine Learning) framework, which is "modular" and therefore open for new extensions, "intelligent" by providing machine learning and deep learning methods on "big data" coming from IoT objects, "server-based" in a service-oriented way by offering services via standart Web protocols. We present an overview of the design and implementation details of MIS-IoT along with a case study evaluation of the system, showing the intelligence capabilities in anomaly detection over real-time weather data.**

Keywords—Internet of things; machine learning; big data analytics; anomaly detection; fault detection;

I. Introduction

Internet of Things (IoT) is a network of Internet connected devices, sensors, and computers. In the literature, IoT is defined as follows: "The Internet of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service" [1][2]. IoT devices are increasing in numbers day by day. U.S. National Intelligence Council states that "by 2025 Internet nodes may reside in things that we use everyday, food packages, furniture, paper documents, and more" [3][2]. These developments lead researchers to develop IoT platforms, frameworks to process and analyze huge IoT sensor data.

Proposed and developed IoT platforms, frameworks have different properties for processing and controlling sensor devices and data. Most of them provide device management and control, sensor data analytics, monitoring real time data, secure communication, data storage. However, there are still some ongoing issues that need to be addressed in this area: interoperability, sensor data analysis, modular machine learning algorithm implementation, failure analysis, ease of development and deployment, open source of platform.

In this study, we present a modular intelligent server-based IoT framework with big data and machine learning (MIS-IoT). Within this structure, IoT framework runs on a server based cloud system that is configurable and controllable through a web based graphical user interface. Among some of the advantages of this IoT framework are software development modularity (modular machine learning algorithms), various ways of input data feed possibilities into the framework (IoT sensor inputs can be provided in different ways) and storing data in big data oriented database. In the proposed design, with modular structure, new implemented machine learning algorithms can be retrieved from database. The other advantage in the developed IoT framework is integrating different sources and different types of IoT sensor data. With modular architecture, different types of data can be used as input. In MIS-IoT, data is acquired with REST, network/socket, semantic and bulk data from files. Meanwhile, another significant problem is storing huge IoT sensor data. In the proposed architecture, sensor data is stored in big data oriented, NOSQL infrastructure: Mongo NOSQL database. Stored sensor data objects in MongoDB can be configured via graphical user interface of framework and streaming sensor data is classified by fine tuning the configuration file.

In order to demonstrate the capabilities of MIS-IoT, we generated a use case where different types of sensor data are collected through different sensory inputs while real-time anomaly detection is implemented through an LSTM deep learning model. The proposed model not only presents the sensor data as a time-series data graph to the user, but also learns the input patterns and detects anomalies within the data stream in real-time and informs the user with appropriate alarms on the system GUI. All of these different capabilities are implemented through integrating the components of the proposed MIS-IoT.

The rest of the paper is structured as follows. The related work is explained in Section II. The overall proposed IoT framework infrastructure is presented in Section III. Use-case scenarios of the proposed IoT framework and the evaluation of finding faulty sensors in IoT networks and also anomaly detection in IoT sensor data are explained and discussed in Section IV. We conclude and give future research directions in Section V.

II. Related Work

IoT world keeps expanding with current technological advancements. IoT sensors and devices started appearing almost everywhere. There is a vast variety of application areas of IoT in our daily lives: Aviation, education, energy, entertainment and sport, environment, finance and banking, food, government, healthcare, home automation, information and communication technologies, logistics, manufacturing, military, retail, transportation, vehicles [2]. In addition, the number of IoT devices which are connected to Internet will be somewhere between 50 to 100 billion [4], [1] within 10 years and the total amount of data generated by humans and devices on earth will reach 35 ZB [2].

In IoT world, there are also different types of research areas: Addressing and naming device, big data, context aware computing, data mangement, device management, energy management, hardware development, information centric network, interoperability and heterogeneity, M2M Communications, machine learning and artificial intelligence, resource management, security and privacy, service management, standardization activities, system and network architecture [2]. IoT platforms, architectures, frameworks and middlewares are emerged from these research areas by researchers from industry and academics.

In addition, IoT standardization activities are important to generalize the proposed and developed IoT platforms, architectures, frameworks. Standardization activities continue to improve for IoT world by OneM2M[1], IETF (Internet Engineering Task Force)[2], ETSI (European Telecommunications Standards Institute)[3] and IEEE (The Institute of Electrical and Electronics Engineers)[4]. OneM2M is a high level standardization activity on IoT devices, working on an interoperability framework towards a common M2M or IoT Service Layer for all types of devices and framework. OneM2M organization have more than 200 members. Some of the biggest organizations are listed as follows: Association of Radio Industries and Businesses (ARIB), Telecommunication Technology Committee (TTC), the Alliance for Telecommunications Industry Solutions (ATIS), Telecommunications Industry Association (TIA), the China Communications Standards Association (CCSA), China; the European Telecommunications Standards Institute (ETSI), and the Telecommunications Technology Association (TTA) [5]. OneM2M IoT ontology design is explained in [6]. IoT standardization activities are surveyed in [2].

IoT platforms, architectures, frameworks and middlewares are proposed and developed by different organizations to handle different problems in the area. Developed IoT platforms, frameworks are used in daily activities by collecting and analyzing sensor data, and controlling IoT devices remotely. There are many proposed IoT platforms, architectures, frameworks and middlewares. Some of the popular IoT platforms are listed as follows: Amazon Web Services (AWS) IoT, Microsoft Azure IoT, Google Cloud Platform, ThingWorx IoT, IBM Watson, Samsung Artik, Cisco IoT Cloud Connect, Hewlett Packard Universal of Things (IoT) Platform, Salesforce IoT Cloud, Datav by Bsquare, Mindsphere by Siemens, Bosch IoT Suite, Carriots, Oracle Integrated Cloud, General Electrics Predix, Arm MBED IoT Device platform, Mosaic (LTI), Mocana, Kaa. In addition, some of the popular IoT architectures, frameworks, middlewares and tools are listed as follows: AllJoyn, AirVantage, Brillo Carriots, Devicehub.net, EvryThng, Ericsson IoT-Framework, Intel IoT Platforms, IoTivity, LinkSmart, OpenMTC, OpenIoT, OpenRemote, Platform.io, Pentaho, realTime.io, Statistica (Dell), SensorCloud, SkySpark, The Thing System, Tellient, ThingSquare, ThingSpeak, Zetta. Proposed IoT platforms, architectures, frameworks are surveyed in [2], [7], [8], [9].

Most of the IoT platforms have the following features: IoT remote device management and control, sensor data analytics, real time data exchange, information monitoring, scalable platform, secure communication, data storage, and rules engine. Even though many IoT solutions are proposed and developed, there are still problems and missing features in this area: interoperability, sensor data analysis, modular machine learning algorithm implementation, failure analysis, ease of use of platform, ease of development, open source of platform, and big data store. Interoperability can be solved with generalized network packet structure. Network packets can be designed and implemented according to IoT standards. Online and offline sensor data analysis capabilities are significant feature for IoT platforms. Statistics and analysis of sensor outputs can provide different advantages such as less energy use, less investment cost. Modular open source machine learning algorithm implementation allows the flexibility of the development of the new and unimplemented machine learning algorithms. Failure analysis can also be considered as a valuable feature. Detecting failed sensor in the IoT network avoids many problems and provides different advantages. In addition, open source IoT platform is important for ease of platform development.

Anomaly detection is a major problem in time series data in many application areas. Malhotra et al. [10] used LSTM for anomaly detection on time series data from the areas of ECG, space shuttle, power demand, and multi-sensor engine datasets. The results indicate up to 90% F-score success rates. Chauhan and Vig [11] have shown anomaly detection in ECG signals for finding different kinds abnormalities in human heart with up to 99% F-score values. Taylor et al. [12] on the other hand worked on the detection of security breaches in automobiles' controller area network using LSTM with varying success rates for different kinds of attacks. All of these and other similar work show that LSTM works better than earlier solutions for anomaly detection in time series data with great success. We also use LSTM in this study for anomaly detection, based on a maximum likelihood analysis.

III. MIS-IoT Framework

Modular Intelligent Server Based IoT Framework (MIS-IoT) is designed with different layers to provide modularity. It has mainly three layers: Data acquisition layer, ETL (Extract, Transform and Load layer, Machine learning and Analytics layer). Semantic module can also be added to framework to transform semantic data. The general conceptual layer design of proposed framework is illustrated in Figure 1. Proposed

[1]http://www.onem2m.org/
[2]https://www.ietf.org/topics/iot/
[3]https://www.etsi.org/technologies-clusters/technologies/internet-of-things
[4]http://standards.ieee.org/innovate/iot/

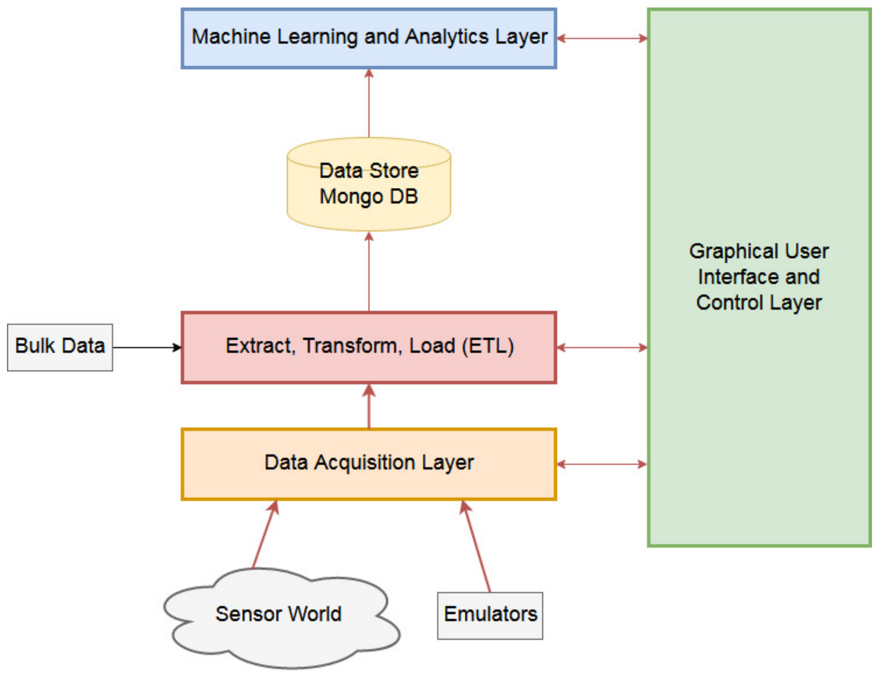

Machine Learning and Analytics Layer
Data Store Mongo DB
Graphical User Interface and Control Layer
Bulk Data
Extract, Transform, Load (ETL)
Data Acquisition Layer
Sensor World
Emulators

Fig. 1. MIS-IoT Framework General Conceptual Design

Framework design structure and implemantation is mentioned in the following sections: Framework design and implementation, server module, front-end module, and machine learning module.

A. *Framework Design and Implementation*

Our proposed framework is designed with different modules: Front-end module, clustered server module and learning module. Front-end module and clustered server module are in the graphical user interface and control layer, while learning module is in the machine learning and analytics layer. Communication interfaces are illustrated in Figure 2. As seen in Figure 2, we used modular approach for implementing the framework. The framework consists of three separate modules as front end module, server module and learning module. All modules communicate with each other using rest architecture. Thus, it is possible to deploy each module on different servers, and with proper rest endpoint settings, the modules can communicate with each other.

B. *Server Module*

Node.js[5] is used for the server module. This module is responsible for saving incoming requests to database, routing learning tasks coming from front end module to learning module, creating live tracking data on specified port for visualization, handling bulkloading tasks and responding query requests for front end usage. The module has been written with Node's clustering techniques for handling simultaneous requests coming from both front end module, socket and rest connections. It automatically creates node sub processes with taking into consideration host's Central Processing Unit (CPU) count. For instance, if the host CPU supports 8 threads, then 8 node processes are created for handling incoming requests.

The server module also supports three different types of communication methods as REST (Representational state transfer) protocol, socket connection and bulkloading. REST is a stateless protocol, therefore there can be nothing saved that has to be remembered by the next transaction. It is a fairly standart protocol and once deployed, any program, no matter what language it is written through, can communicate with each other using REST API (that uses HTTP requests with get, put, post, patch, delete HTTP methods). Therefore we added rest interface to the server module. As an alternative to REST, socket connection also can be used. Unlike REST, socket connection is bi-directional and there is no request and response. Once the connection is opened, the server and the client can communicate with each other. There is no need to open new connection for each message. That is the main advantage over the REST architecture. REST can cause overhead on system as it needs to open connection with each request. Therefore, we decided to employ socket support on our framework as an alternative to REST architecture. The module also supports bulkloading for reading and importing lots of Comma-Seperated Value (CSV) files at once to the system. It is managed by the front end module, and once the path is specified, the folder is processed and files that have CSV extensions are loaded to the database.

We used MongoDB[6] for storing purposes. NoSQL is more convenient for large sensor applications which tend to grow horizontally [13]. The framework needs simple configuration file for deciding how to classify incoming data from any source. Regardless of the chosen connection type to framework as we mentioned in previous sections, all incoming data is classified by the framework by using a configuration file. For now, the configuration file should be included in the server module file directory. Later, it will be configurable by using

[5]https://nodejs.org/en/

[6]https://www.mongodb.com/

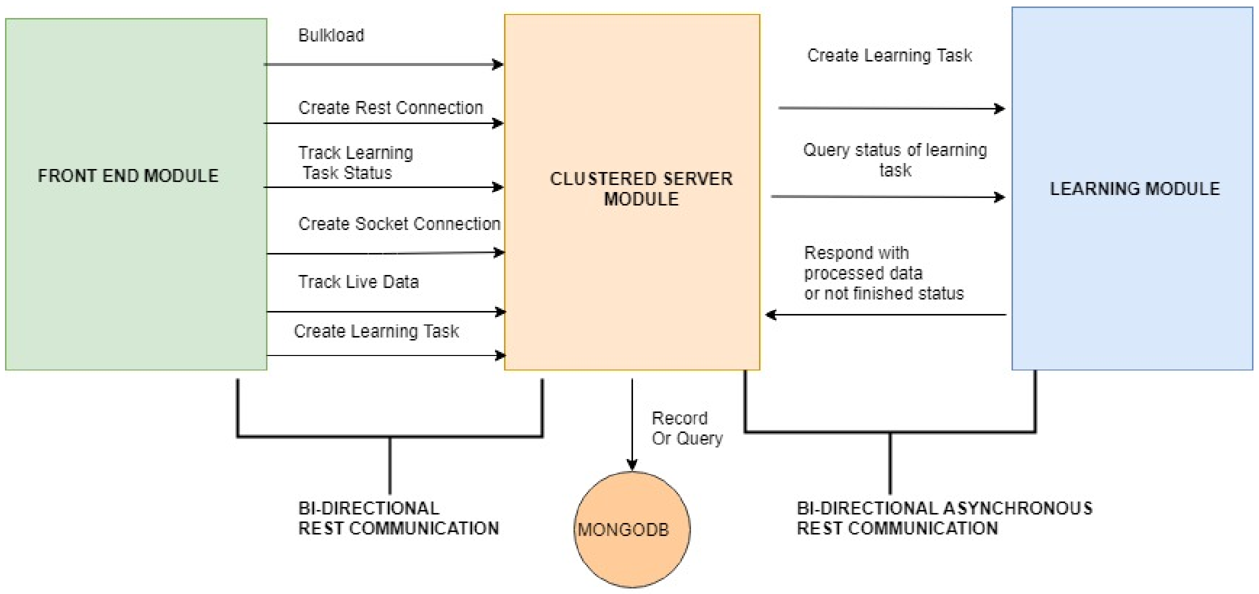


Fig. 2. MIS-IoT Framework Modules and Their Communication Interfaces

GUI.

C. Front-End Module

Vue.js[7] is used for the front end module. Recently its popularity has grown and it is a viable strong to other frameworks such as Angular.js and React.js. We employed Vue.js because we think it has an easier learning curve than others and with relatively less effort, it is possible to produce more. Also, Vuetify.js [8] is used as the material design library to prevent spending unnecessary time on visualization.

The front end module is responsible for managing the server module. It is capable of creating and destroying connections. The rest or socket connection can be created on the specified port. It has abilities for tracking the specified port's data flow. It can create learning tasks and watch their status. Bulkloading can also be done via this module. In short, it is designed to manage the server without too much hassle. Therefore, once the server and learning modules are deployed, it is manageable from the front end module.

D. Machine Learning Module

The learning module is responsible for learning tasks and is written in Python [9]. Scikit-learn[10], Keras[11] and Tensorflow[12] libraries are used to implement clustering and deep learning models. The module supports rest communication and communicates with the server module for creating learning tasks. Since training a model can consume significant amount of time, we designed this module with an asynchronous approach. The server module creates and sends the learning task request to the learning module and assigns an ID for tracing the status of the request.

In the proposed framework, machine learning models and methods are used to find sensor faults and find anomalies on historic sensor data. The proposed IoT framework has an advantage over the other IoT frameworks by implementing modular machine learning and deep learning methods. Modular development architecture is created with retrieving data from database using functions that are used as application programming interface (API).

To find the faulty sensors between a large number of sensors, clustering algorithms are used. Clustering algorithms are classified as unsupervised learning models. In our proposed framework, K-means clustering algorithm is implemented to find faulty sensors in a large number of sensors. With K-means algorithms, it is ensured that the similarities within the cluster are maximum and the similarities between the clusters are minimum. Large scale data can be clustered quickly and efficiently with K-means algorithm [14],[15]. "K" represents the number of fixed clusters needed before starting the algorithm. With the repeated partitioning structure, the K-means algorithm reduces the sum of the distances to the cluster to which each data belongs. K-means algorithm is an iterative algorithm. It first selects the center points of the clusters randomly. Then, the euclidean distances between each point and center points are calculated. New center points are recalculated according to the minimum distance of each point to its nearest cluster center. Equation 1 illustrates the way of finding new cluster centers (C_i: center point of i^{th} cluster, n_i: number of data points in that cluster, X_i: data points). It continues iteratively. If no center point is reassigned again, algorithm is terminated. The algorithm is aimed to minimize the objective function $J(C)$ (Equation 2: C_i: center point of i^{th} cluster, n_i: number of data points in that cluster, X_i: data points, n: number of the cluster

[7]https://vuejs.org/
[8]https://vuetifyjs.com/en/
[9]https://www.python.org/
[10]http://scikit-learn.org/
[11]https://keras.io/
[12]https://www.tensorflow.org/

centers, $||X_i - C_i||$: euclidean distance between X_i, C_i)

$$C_i = \frac{1}{n_i}\sum_{j=1}^{n_i} X_i \tag{1}$$

$$J(C) = \sum_{i=1}^{n}\sum_{j=1}^{n_i}(||X_i - C_i||)^2 \tag{2}$$

In our implementation, instant sensor values are used as input and these sensor values are clustered. In the framework, more than one type of sensor values also clustered according to selection of the different types of sensors. Implemented finding faulty sensor algorithm is expressed in Algorithm 1.

To find anomalies and outliers in the historic sensor data, recurrent neural networks (RNN) algorithm especially long short term memory (LSTM) is used. LSTM [16] models are mostly used with time series data. It is used in different applications such as natural language processing, language modelling, speech recognition, sentiment analysis, predictive analysis, etc. In our proposed framework, LSTM model is implemented to find anomalies and outliers in historic sensor data. LSTM models consist of LSTM units. LSTM unit is composed of cells having input gate, output gate and forget gate. Three gates regulate the information flow. With these features, each cell remembers the desired values over arbitrary time intervals. LSTM cells combine to form layers of neural networks. Equations 3, 4, 5, 6, 7 illustrate the compact form of the forward pass of the LSTM unit [16] (x_t: input vector to the LSTM unit, f_t: forget gate's activation vector, i_t: input gate's activation vector, o_t: output gate's activation vector, h_t: output vector of the LSTM unit, c_t: cell state vector, W , U: weight matrices that need to be learned, b: bias vector parameters that need to be learned, σ_g: sigmoid function, σ_c , σ_h: hyperbolic tangent function, $*$: element-wise (Hadamard) product).

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{3}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{4}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$

$$c_t = f_t * c_{t-1} + i_t * \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{6}$$

$$h_t = o_t * \sigma_h(c_t) \tag{7}$$

In our implementation, historic sensor values of one type of sensor are retrieved from database and they are used as input to LSTM network. A layer LSTM network that composed of 25 neurons, is trained with raw-sensor values (input layer: 1 neuron, hidden layer: 25 neurons, output layer: 1 neuron, dropout: 0.1, epoch: 1000, time steps: 240, optimizer: adam, loss: mse, batch size: 240) [17], [18]. Hyperparameter values are tuned with experimental tests. After prediction of the next value of sensor values, differences of actual and predicted values are determined. The average and standard deviation of the list of difference of actual and predicted values are calculated. Finally, two-sigma and three-sigma values are labelled as anomalies/outliers. Implemented finding anomalies in historic sensor values algorithm is expressed in Algorithm 2.

IV. Use-Case Scenario and Evaluation

A. Configuration

In this use-case scenario, we used LinkedSensorData and LinkedObservationData[13] which contain different weather sensors such as air temperature, wind speed, relative humidity, pressure, visibility, etc. LinkedSensorData is a Resource Description Framework (RDF: standard model using data interchange on the web) dataset that defines approximately 8000 weather sensors information (ex: latitude, longitude, type, etc.) in United States. The example configuration file is as follows:

```
<collection>
    <collectionName>Weather</collectionName>
    <type>
        <typeName>Coordinate</typeName>
        <columns>lat,longt,alt</columns>
    </type>
    <type>
        <typeName>WindInfo</typeName>
        <columns>windGust,windDirection,
                 windSpeed</columns>
    </type>
</collection>
```

We converted RDF dataset to CSV files by using the semantic module. The semantic module is responsible for parsing RDF dataset and producing CSV files. We have one collection named as Weather. Weather collection consists two types of data respectively; Coordinate and WindInfo, and also these data types consist of several fields. When the framework receives JavaScript Object Notation (JSON) data from connection sources, the framework classifies that data by finding the most intersected columns. The __type field is added to document object and it is saved to related collection with that type. For example, if we assume that the incoming data consists of these fields:

```
{
"windGust" : 36,
"windDirection": 37.45
"windSpeed": 48
}
```

These fields are mostly intersected with WindInfo type, so it is classified as WindInfo (__type:WindInfo is added into document) and saved to Weather collection. The end result will look like:

```
{
"__type": "WindInfo"
"windGust" : 36,
"windDirection": 37.45
"windSpeed": 48
}
```

[13]http://wiki.knoesis.org/index.php/LinkedSensorData

Algorithm 1 Finding Faulty Sensors

```
1: function FINDFAULTYSENSORS()
2:     Input: $numberOfSensorTypes, sensor\ data\ from\ database$
3:     Output: $table[], plots$
4:     $Retrieving\ sensor\ data\ from\ database\ and\ convert\ to\ 'file.csv'$
5:     $dataFrame = pandas.readcsv('file.csv')$
6:     $table = dataFrame.pivottable(index = ["Name"])$
7:     $table = table.dropna()$
8:     $if(numberOfSensorTypes == 1):$
9:         $no\ need\ to\ normalize\ sensor\ data$
10:    $elif(numberOfSensorTypes == 2):$
11:        $tableNormalized = normalize2values(table)$
12:    $elif(numberOfSensorTypes == 3):$
13:        $tableNormalized = normalize3values(table)$
14:    $k = numberOfClusters$
15:    $cluster = sklearn.cluster.KMeans(nclusters = numberOfSensorTypes)$
16:    $if(numberOfSensorTypes == 1):$
17:        $table["Cluster"] = cluster.fitpredict(table[table.columns[:]])$
18:    $else:$
19:        $table["Cluster"] = cluster.fitpredict(tableNormalized[tableNormalized.columns[:]])$
20:    $plotResults()\ in\ terms\ of\ Clusters,\ Sensor\ Positions(Latitude, Longitude)\ using\ MatplotLib$
21:    $Store\ faulty\ sensors\ in\ terms\ of\ Clusters,\ Sensor\ Positions(Latitude, Longitude),\ SensorValues$
```

The framework supports REST and socket connection methods for data input. As seen in Figure 3, REST or socket connection can be initiated for the desired ports. After configuring the ports, the framework handles incoming request in the specified ports. We regularly sent the specified request to the framework and created few records for demonstration purposes.

B. Monitoring: Online IoT Sensor Values Display

Real time port tracking also can be made for specified port from GUI as seen in Figure 4. In this case, we can see that request to specified port consists of windGust, windDirection and windSpeed fields.

C. Learning: Anomaly Detection in Historic IoT Sensor Values

Since we have records in database, we can now create learning tasks from existing fields as seen in Figure 5. Learning tasks can be created step by step. As a first step, the user chooses the available collection in the database. In this case, since the classified type belongs to Weather collection, we chose weather. Secondly, the user selects the available types. One collection can have multiple types. Thereafter, the user chooses from the available fields in the Weather collection and WindInfo type. In our case, windGust, windDirection and windSpeed are the possible options. Selected field values are being aggreageted.

In the fourth step, the user decides LSTM algorithm parameters and learning request is created for cron job in order to send the request to learning module. The server module creates and sends learning task request to the learning module and gives it an ID for tracing the status of request. In this step, the request can have four status determined by us as *to_be_processed*, *processing*, *done* and *failed*. If the learning request is created for the first time, its status is marked as *to_be_processed*. A regular cron job which works on server module in every one minute and sends requests that have *to_be_processed* status to the learning module for initiating the learning task. If the server receives a successful response, it changes the status of the request to *processing*. The cron job also has responsibility for tracking the requests with processing status. It creates query with the tracking ID and sends REST request to the learning module in order to track the current status of learning process. If the learning module returns a successful response along with processed data, the cron job sets the status of the request as *done*. At this point, the processed data can be visualized by the front end module. If it takes longer than a threshold parameter determined by user and status is still processing, the cron job marks the request as *failed*. Failed tasks can be retried by GUI if the user choose the retry option from the GUI. A learning request example is as follows:

```
{
_id : ObjectId("5b7202480caa88360c0c1a5b"),
request: {
            type: WindInfo,
            field: windgust,
            collection: Weather,
         },
aggreagetedFieldValues: [91.277, 91.150,
   91.29, 116.01, 119.56],
status: TO_BE_PROCESSED,
__v: 0
}
```

We used 19764 samples of air temperature data which belongs to Katrina hurricane. We weeded out N/A values and executed the algorithm on remaining dataset. After that, LSTM algorithm was run to predict values. LSTM training error was 8.66 RMSE (Root Mean Square Error) and LSTM test error was 17.25 RMSE. In labelling anomaly phase, as a first step, we created a list that holds differences between original data and corresponding prediction values and calculated the mean

Algorithm 2 Finding Anomalies in Time-series Data from Sensors

```
function FINDANOMALIES()
    Input: $sensor\ data\ from\ database$
    Output: $table[], plots$
    $Retrieving\ sensor\ data\ from\ database\ and\ convert\ to\ file.csv$
    $dataFrame = pandas.readcsv('file.csv')$
    $table = dataFrame.pivottable(index = ["Name"])$
    $table = table.dropna()$
    $scaler = MinMaxScaler(feature\ range = (0, 1))$
    $table = scaler.fitTransform(table)$
    $trainSize = int(len(dataset) * 0.5)$
    $testSize = len(dataset) - train\ size$
    $train, test = dataset[0 : train\ size, :], dataset[trainSize : len(dataset), :]$
    $trainX, trainY = train.iloc[:, : -1], train['label']$
    $testX, testY = test[].iloc[:, : -1], test['label']$
    $model = Sequential()$
    $model.add(LSTM(25, inputShape = (1, time\ steps = 240)))$
    $model.add(Dropout(0.1))$
    $model.add(Dense(1))$
    $model.compile(loss =' mse', optimizer =' adam')$
    $model.fit(trainX, trainY, epochs = 1000, batchSize = 240, verbose = 1)$
    $trainPredict = model.predict(trainX)$
    $testPredict = model.predict(testX)$
    $trainPredict = scaler.inverseTransform(trainPredict)$
    $testPredict = scaler.inverseTransform(testPredict)$
    $Differences\ of\ each\ actual\ and\ each\ predicted\ values\ are\ calculated\ and\ create\ list$
    $Average\ and\ standard\ deviation\ of\ the\ list\ are\ calculated$
    $Two - sigma\ and\ three - sigma\ values\ are\ labelled\ as\ anomalies/outliers$
    $plotResults()\ in\ terms\ of\ sensor\ values\ and\ anomalies\ using\ MatplotLib$
    $Store\ anomalies\ in\ database$
```

and standart deviation (sigma) values of the list. In our case, sigma was 11.37 and mean was 9.46. Thereafter, we calculated two and three sigma values and used it as a threshold for whether classifying instance as an anomaly or not. As seen in Figure 6, we found 146 anomalies on 9641 test records with three sigma limit. Anomaly points are labelled as red points. We also tried with two sigma limits as threshold and found 287 anomalies.

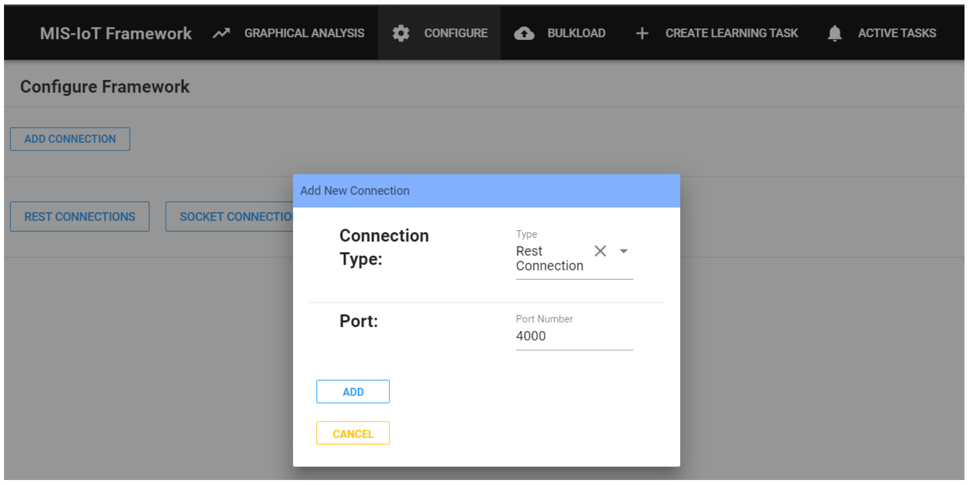
D. Learning: Fault Analysis for IoT nodes

In our framework, we also found the faulty sensors in the IoT network. In the learning module, instant sensor data values from IoT networks are clustered for faulty sensor detection. When we performed clustering analysis on the wind speed and relative humidity alone, some sensor data results were substantially different from their neighboring regions. We provided an example use case for detecting the faulty sensors. Figure 7 illustrates the instant relative humidity sensor values clustering analysis [19]. Figure 8 shows the instant wind speed sensor values clustering analysis [19]. Faulty sensor values can be seen in the form of ”X”. In that particular case the faulty sensors form separate clusters, far away from the other data points. As a result, for these faulty sensors, the number of clusters mostly do not prevent the detection of the faulty sensor as such, they always form a separate cluster, since their data readings are very different from the others. In addition, details of faulty analysis use-case scenario are also mentioned in [19].

E. Discussion

In our study, we wanted to present how anomaly detection could be applied for any given problem. For that we analyzed 2 separate use cases. In the first one we used k-means to cluster the data into a number of clusters depending on the particular implementation requirements. For example when the Relative Humidity is divided into 3 clusters (Figure 7), one of the clusters only had one data point coming from a particular sensor. All its neighboring sensors had significantly different (can be low or high) readings, so our assumption was the sensor was faulty. Similar analysis can be done for Wind Speed Clustering illustrated in Figure 8. We wanted to demonstrate how we were able to utilize k-means through our framework for faulty sensor detection. Meanwhile, the users should check the applicability and adaptability of k-means for their particular implementation case, since with large number of k, the algorithm might result in empty clusters [20] which might cause instabilities for the anomaly detection. For our use case, we ignored this scenario. The ”faultiness” rules can be adjusted or defined by the user for any intended purposes. As a matter of fact, the second use-case demonstrates how that can be done through another learning algorithm.

In that particular case, we used the air temperature data and defined the anomalies as ”data points that lie outside 2 or 3 sigma levels”. Hence, we labeled these points as anomalies within the training set. Once, the learning phase was finished, the corresponding LSTM network successfully detected the anomalies within the test set, which obviously were not



Configure Framework

Add New Connection

Connection Type: Type: Rest Connection

Port: Port Number: 4000

Fig. 3. Configuration

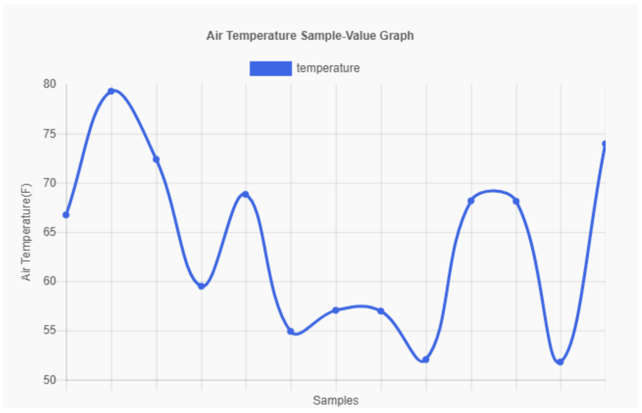
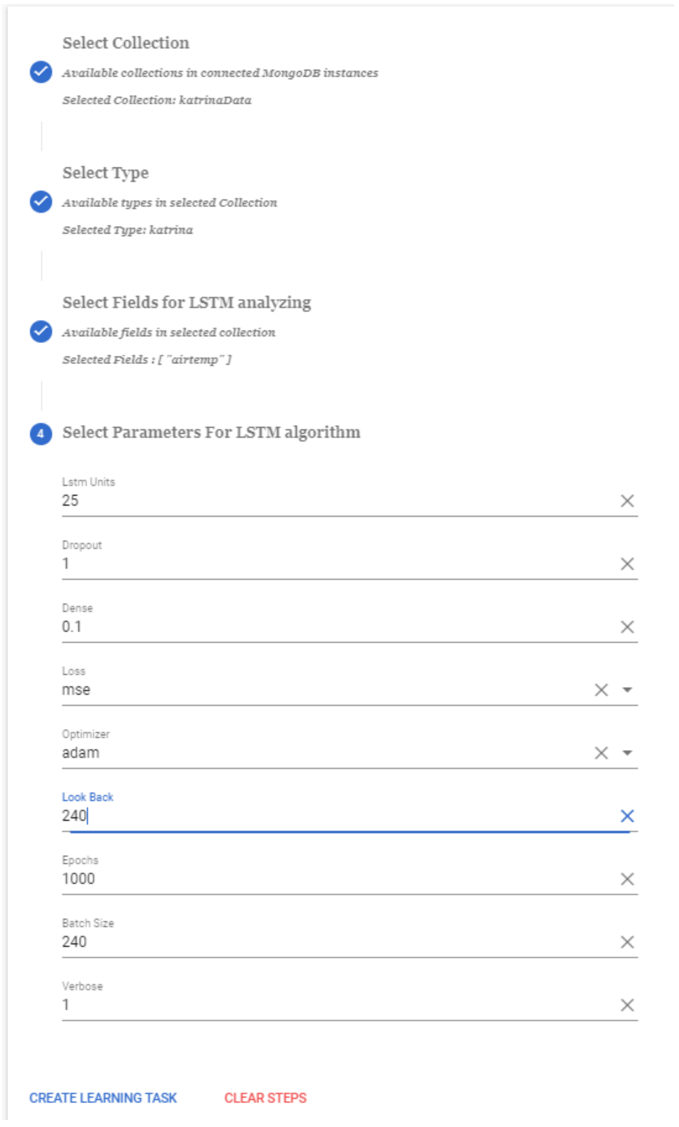


Fig. 4. Online IoT Sensor Values Display



Select Collection

Available collections in connected MongoDB instances

Selected Collection: katrinaData

Select Type

Available types in selected Collection

Selected Type: katrina

Select Fields for LSTM analyzing

Available fields in selected collection

Selected Fields : ["airtemp"]

4 Select Parameters For LSTM algorithm

Lstm Units: 25

Dropout: 1

Dense: 0.1

Loss: mse

Optimizer: adam

Look Back: 240

Epochs: 1000

Batch Size: 240

Verbose: 1

Fig. 5. Creating Learning Task of Anomaly Detection in Historic IoT Sensor Values

labeled. We observed that the model was successful, such that the points that were detected as anomalies were in fact 2 or 3 sigma apart from the mean. In any given implementation, the user can define their rules for classification, prediction or anomaly detection. Our main motivation was to demonstrate the possibilities for customizing tailor-made training rules for the learning models introduced to the framework.

V. Conclusion

We presented a modular IoT framework (MIS-IoT) that can be easily expanded with data acquisition, learning, and analytics capabilities depending on the particular needs of the associated IoT environments. In the particular use case we presented in this paper, we tested our proposed framework in weather data analysis and faulty sensor detection problem. We simulated the acquisition of streaming weather sensor data and developed a real-time anomaly detection method using LSTM deep learning model and K-means clustering. The results indicate that our framework can be used in environments with different sensors and various data capabilities in which data need to be integrated for predictive analytics in real-time. For future work, we are planning to use the framework in a machining process requiring different sensor data (vibration, etc.) for a possible Industry 4.0 implementation.

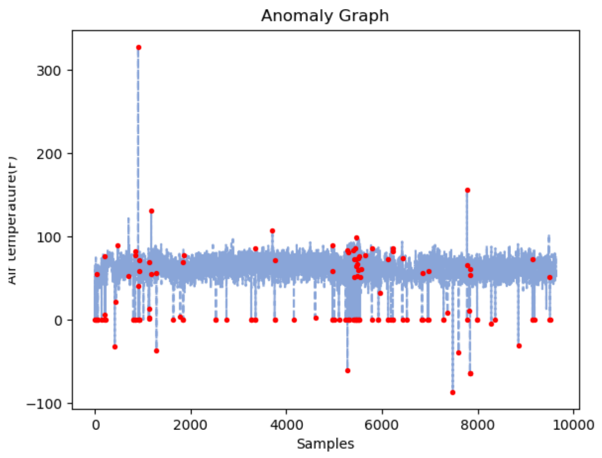


Fig. 6. Anomalies with Three Sigma Limits

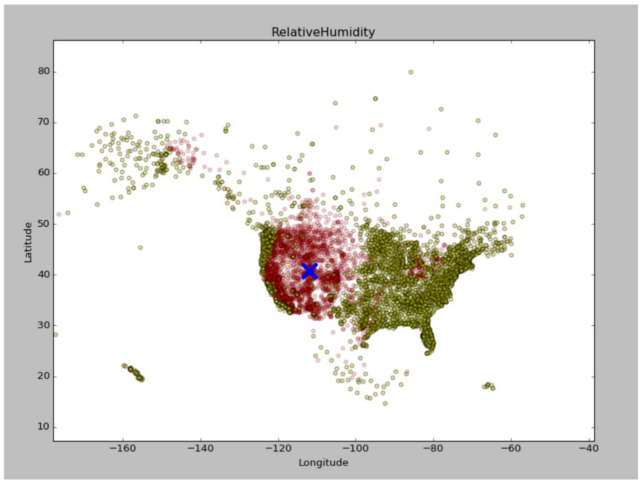


Fig. 7. Relative Humidity 3 Clusters, Sensor Fault [19]

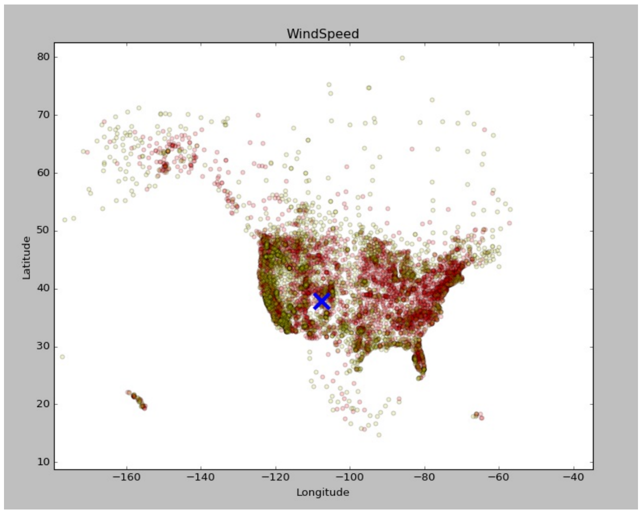


Fig. 8. Wind Speed 3 Clusters, Sensor Fault [19]

References

[1] C. Perera, A. Zaslavsky, M. Compton, P. Christen, and D. Georgakopoulos, "Semantic-Driven Configuration of Internet of Things Middleware," in *9th International Conference on Semantics, Knowledge and Grids*, IEEE, 2013, pp. 66–73.

[2] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in internet of things: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 1–27, 2018.

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[4] O. B. Sezer, E. Dogdu, M. Ozbayoglu, and A. Onal, "An extended iot framework with semantics, big data, and analytics," in *Big Data (Big Data), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1849–1856.

[5] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 20–26, 2014.

[6] M. B. Alaya, S. Medjiah, T. Monteil, and K. Drira, "Toward semantic interoperability in onem2m architecture," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 35–41, 2015.

[7] C. Perera, C. H. I. H. Liu, S. Jayawardena, and M. Chen, "A Survey on Internet of Things From Industrial Market Perspective," *IEEE Access*, vol. 2, no. 2014, pp. 1660–1679, 2014.

[8] C. Perera, C. H. Liu, and S. Jayawardena, "The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 4, pp. 585–598, 2015.

[9] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Computer Communications*, vol. 89-90, no. 9, pp. 5–16, 2015.

[10] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, Presses universitaires de Louvain, 2015, p. 89.

[11] S. Chauhan and L. Vig, "Anomaly detection in ecg time signals via deep long short-term memory networks," in *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, IEEE, 2015, pp. 1–7.

[12] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 130–139.

[13] J. S. van der Veen, B. van der Waaij, and R. J. Meijer, "Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual," in *2012 IEEE Fifth International Conference on Cloud Computing*, IEEE, Jun. 2012, pp. 431–438.

[14] J. Karimov and M. Ozbayoglu, "Clustering quality improvement of k-means using a hybrid evolutionary model," *Procedia Computer Science*, vol. 61, pp. 38–45, 2015.

[15] J. Karimov, A. M. Ozbayoglu, and E. Dogdu, "K-means performance improvements with centroid calculation

heuristics both for serial and parallel environments.," in *BigData Congress*, 2015, pp. 444–451.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] O. B. Sezer and A. M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, 2018.

[18] T. Fischer and C. Krauß, "Deep learning with long short-term memory networks for financial market predictions," FAU Discussion Papers in Economics, Tech. Rep., 2017.

[19] A. C. Onal, O. B. Sezer, M. Ozbayoglu, and E. Dogdu, "Weather data analysis and sensor fault detection using an extended iot framework with semantics, big data, and machine learning," in *Big Data (Big Data), 2017 IEEE International Conference on*, IEEE, 2017, pp. 2037–2046.

[20] J. Karimov and M. Ozbayoglu, "High quality clustering of big data and solving empty-clustering problem with an evolutionary hybrid algorithm," in *Big Data (Big Data), 2015 IEEE International Conference on*, IEEE, 2015, pp. 1473–1478.