# A smart city dashboard for combining and analysing multi-source data streams

**DOI:**
10.1109/HPCC/SmartCity/DSS.2018.00226

# A smart city dashboard for combining and analysing multi-source data streams

Ann Gledson, Thamer Ba Dhafari, Norman Paton and John Keane
School of Computer Science, University of Manchester, UK
Email: {first-name}.{last-name}@manchester.ac.uk

***Abstract*—Smart city projects are making available numerous, diverse, potentially valuable data streams. However, it is cumbersome and laborious for domain experts to identify, analyse and inter-relate such streams. In this paper we describe a system that is designed to support these users to obtain an understanding of key features of individual streams, and to undertake a variety of analyses that inter-relate multiple streams. Three case studies are presented, showing the use of the system to process live car-park, weather, building energy consumption and bicycle journey data from the CityVerve Smart City project in Manchester.**

## I. Introduction

Smart city projects collect dynamic data with a view to supporting service delivery, providing timely information to users, and guiding longer-term decision making. Such projects utilise real-time, streamed Internet of Things (IoT) data from a variety of domains, such as healthcare, air quality and transport. To address these challenges, [1] introduce the *sensing as a service* model comprising four layers: *sensors and sensor owners*, *sensor publishers*, *extended service providers* and *sensor data consumers*. The dashboard we describe provides an *extended service*, allowing *sensor data consumers* to easily view, manipulate and analyse the published data streams.

In the Smart City context, the *sensor owners* and *sensor publishers* are many and largely independent [2], resulting in non-standardised and highly heterogeneous data. Solutions mainly involve either proactive and collaborative adherence to pre-defined data description standards, as suggested in [3][1], or retroactive development of individual tools to integrate such datasets [4][5][6]. The dashboard presented in this work takes advantage of the available Hypercat standard [7][8] for searching data hubs, but adopts the latter approach for fetching data points, as each hub exposes these datapoints in a different format. In addition, IoT data is mobile and dynamic, both spatially and temporally [9], and the analysis of such information demands dynamic and flexible approaches; in contrast to traditional *store-then-process* [10] solutions [3]. Such *on-the-fly* [10] data processing allows consumers to explore [11][12], manipulate and integrate [9] sensed data as as it becomes available, allowing *the communication of critical change rather than the communication of full history* [13].

Potential users of smart city data are numerous and diverse [14][15]; they include domain experts such as town planners [16][10][17] and environment/healthcare informaticians [18][9], as well as stakeholders living or working in the city [16]; each views the data from their own experiences and perspective. Enabling a diverse user set to explore multi-modal cross-thematic data increases the potential to find useful patterns to meet a city's needs. To support users in traversing this complex information space, smart city projects usually provide one of three approaches:

1) *Data dashboards* typically bring together several mash-ups to provide custom visualisations of predefined data sets (e.g. www.cityofboston.gov/mayorsdashboard and www.dublindashboard.ie) [19][20][21]. Such dashboards are useful, but the list of datasets provided is static and the facilities for the user to explore, process and analyse *additional* datasets is limited. For example, Dublin City's dashboard presents a range of real-time data such as current weather, drive times and house prices, as well as static documents such as planning applications and city maps. The data follows a typical web-site structure, by which the user can select *predefined* menu/list items to drill down to the lowest level required.
2) *Data Stream Management Systems* (DSMS) support continuous and *ad hoc* queries over multiple data streams [22][23][24][25]. These systems are often for private use within individual institutions rather than providing open data access. Furthermore, the required continuous queries can range from simple analysis tasks, for example computing aggregate operations over windows to complex analyses requiring a significant level of computing skills. For such tasks a user is likely to have to resort to transferring the data to a data analytics software package, such as Python Pandas or R.
3) *Data Extraction and datastream analytics*: Data extraction tools allow exploration and extraction of published online data; for example [26] describe a tool *for real-time extraction of real-world data* of use *for research and development*; while [5] present an open data search system which supports *ad hoc*, interactive discovery of connections or *linkages* between datasets. These systems output files of tabular data that the user can store and analyse in a static manner, using external data analysis tools. Much work has also been done on *datastream analytics*, including pre-processing [27], analysis [9] [28] and on data mining IoT data specifically [29] [17].

The cross-theme dashboard described here, developed as part of the CityVerve Smart City project in Manchester, synthesizes aspects of all the above approaches, creating a single tool to support interactive data exploration, extraction, visualisation, transformation and *ad hoc* analysis of publicly available IoT streamed data. It amalgamates independently published data sets from different domains and provides insights into potential utility of city data, with relatively modest levels of effort.

To illustrate this functionality, we introduce three analysis case studies:

1) an evaluation of the hypothesis that rainfall affects the decision to drive into the city centre on work-days;
2) a comparison of weather data along with energy usage in two University of Manchester buildings; and
3) an overview of city centre cycling, using data from a live bicycle trial (https://seesense.cc/pages/manchester) to obtain an overview of bicycle usage over a 25 day period and at different hours of day.

These tasks will include searches for the relevant Manchester city centre data streams, their manipulation and the analysis of the processed data using correlation, time series analysis and hierarchical clustering.

## II. Technical Context

### A. Overview of the CityVerve Project

The CityVerve project is intended to combine the latest IoT technologies to *demonstrate the capability of IoT applications and address barriers to deploying smart cities, such as city governance, user trust and adoption, interoperability and scalability* (http://www.cityverve.org.uk). The project covers four areas: transport and travel, health and social care, energy and the environment, and culture and the public realm. The data architecture is based on a centralised *platform of platforms* (PoP), a technology layer that provides a secure catalogue of available data, and which has been developed to accommodate the numerous and evolving data publishers in the city. The PoP is designed to act as a single point of access, using data virtualisation to provide seamless access to the underlying data hubs: the BT CityVerve Data Hub (transport and environment data)[8] (https://portal.bt-hypercat.com) , the Asset Mapping hub (https://www.assetmapping.com) (data relating to physical assets belonging to buildings and estates) and the DataWell hub (anonymised health data). These hubs incorporate the results from both new and existing sensing capabilities from multiple data providers.

A key component of the architecture is a catalogue that follows the Hypercat specification [30]; this allows a data consumer to discover information about sensor data assets over the web. Similarly, the sensor data streams are exposed using publicly available and RESTful APIs that provide easy access for application developers. Adherence to such specifications has facilitated a dashboard design allowing searching of *any* external (non-CityVerve) hub catalogue that follows this Hypercat specification and the fetching of datapoints belonging to each data feed requires only minor information about the hubs required (RESTful) URI format. For example, a further external hub from the EU Triangulum project (http://triangulum-project.eu) has been seamlessly assimilated into the dashboard and the case studies include data from this *external* hub.

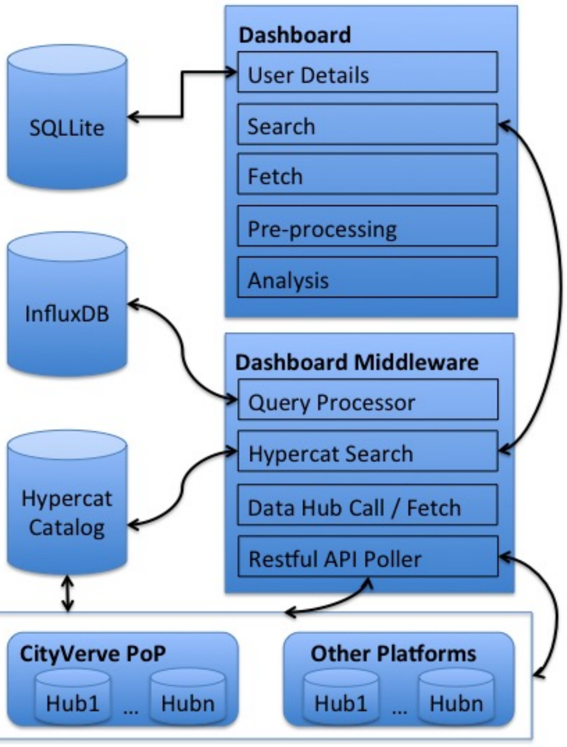


Fig. 1. Dashboard Architecture.

### B. Dashboard Architecture

The CityVerve data dashboard architecture is illustrated in Figure 1; we summarize its key components.

The *middleware* layer links the dashboard and the data hubs. It comprises four main components:

1) *Stream Search*: the *Hypercat search* component searches the data hub catalogues given a single curl (https://curl.haxx.se) query string following the Hypercat specification [7] and a list of identifiers for Hypercat compliant hubs. Non-PoP hub catalogues are searched using BT's on-line API (http://search.bt-hypercat.com).
2) *Data call/fetch*: the data hub *Call/Fetch* library incorporates the connectors required to build request URIs for fetching data-points from the restful data hub APIs and to handle the results; the current hubs available are the CityVerve PoP hub, the BT data hub[8] and the Triangulum project hub.
3) *Polling and Storage*: the *Restful API poller* component uses the above library to continually poll the selected data streams; the returned data is stored in an InfluxDb (https://docs.influxdata.com) time-series database (DB); InfluxDb supports: (i) fast ingestion of time series data; (ii) the addition of extra stream information using indexable data tags (such as the URI of the original stream and geographical location data); and (iii) an SQL-like *Continuous Query (CQ)* language allowing filtering and aggregation of the raw data-points.
4) *Query Processor*: converts user selected data manipula-

tion requests into the necessary continuous queries and the resulting streams are stored in the InfluxDb.

Building on the middleware layer, the *dashboard* component is a web application built using the Django framework. The GUI consists of separate web-pages for the search, manipulation and analysis of data. The dashboard also contains a *user details* component that stores data in an SQLight DBMS.

To explain the *Hypercat search* process, we firstly define data feeds and data streams along with how these are expressed in the Hypercat data catalogue [7][30]. A *data feed* is a collection of one or more *data streams*. For example, a sensor attached to a lamppost and measuring both oxygen and nitrogen dioxide or a road traffic incident reporting mechanism are examples of data feeds. Each will have the property *hasSensorStream* and may have further properties such as *latitude* and *longitude*. A *data stream* is a series of individual measurements, such as nitrogen oxide, oxygen or accident severity. It will have the *belongsToSensorFeed* metadata property and may have further properties such as *minValue*, *maxValue*, *location* and *unitSymbol* (an example value being *degrees Celsius*). These *data streams* represent the individual values in our dashboard; feed metadata are used only to obtain higher-level information about the streams, such as the type of streams held or a location. A Hypercat catalogue includes a list of all feeds and streams and their lowest level descriptors are expressed in *relation:value* pairs such as:

```
{"rel":"urn:X-hypercat:rels:hasDescription:en",
 "val":"Imperial sensor data."}
```

The Hypercat specification also includes an expressive query language that combines sets of operands with any number of nested UNION and INTERSECTION operators and each operand (or *query*) is also specified using the above relation-value structure. For example:

```
"intersection": [
  {"query": "?rel=urn:X-bt:rels:feedTag"},
  {"query": "?val=Air Quality"}]
```

would return all feeds and streams with a relation *feedTag* which has the value of *Air Quality*. Geographical (bounding box) searches are also provided, for example:

```
{"query": "?geobound-minlat=53.46
   &geobound-maxlat=53.48
    &geobound-minlong=-2.26
    &geobound-maxlong=-2.22"}
```

would return feeds within the minimum and maximum latitude and longitudes shown. The dashboards functionality is to expose the middleware and Hypercat query functionality to the user in an accessible, graphical format, as illustrated below.

## III. Dashboard Functionality

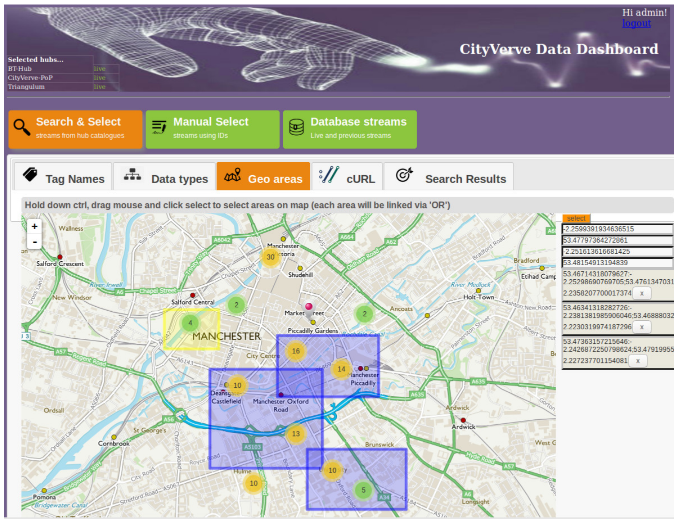
As highlighted in Section 1, *to make use of smart city data, users need to be able to easily manipulate streamed data that is dynamic, heterogeneous (from diverse sources and representing many types of information) and non-standardised.* In the previous section we presented the middleware layer that contributes towards handling these requirements so that we are able to present a uniform set of streams on the dashboard that can be continually updated and imported into a database. The challenge remains to design a dashboard application that allows users to view many streams, select a subset of those for polling and to provide the key data manipulation and data analysis/visualisation techniques in an accessible manner.

### A. Stream states

Given the search mechanism, the polling process, the storage of streams in InfluxDB and the ability to create derived streams, each stream is in one of 4 states:

1) *Unused*: neither currently being polled nor held in InfluxDB;
2) *Previous*: previously selected by the user and still held in the database for persistence reasons;
3) *Live*: currently being polled and stored in the database;
4) *Derived*: currently in the database and resulting from pre-processing steps. For example *precipitation* aggregated by hour and taking the mean value.

### B. Searching for Streams



Selected hubs...

| | |
|---|---|
| BT-Hub | live |
| CityVerve-PoP | live |
| Triangulum | live |

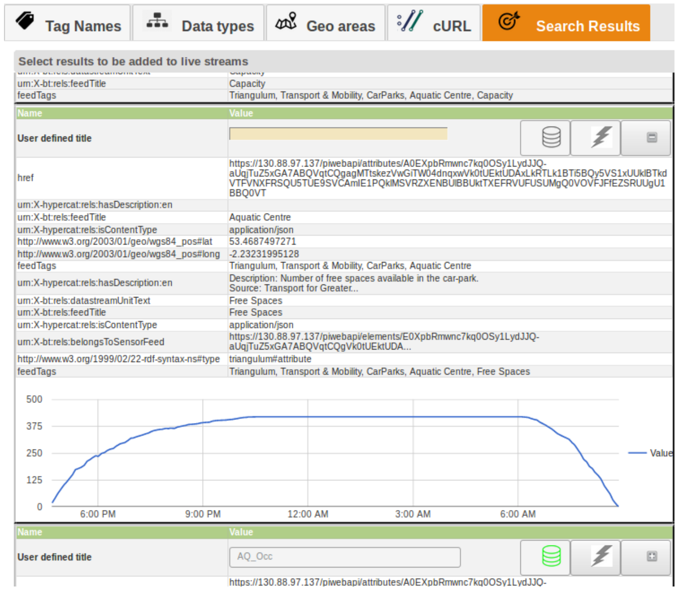
Hold down ctrl, drag mouse and click select to select areas on map (each area will be linked via 'OR')

-2.2599391934636515
53.47797364272861
-2.251613616681425
53.48154913194839
53.46714318079627:-2.25298690769705;53.4761347031 2.2358207700017374
53.46341318282726:-2.2381381985906046;53.46888032 2.2230319974187296
53.47363157215646:-2.2426872250798624;53.47919955 2.227237701154081

Fig. 2. Search/Fetch page: geographical search

Once the user has logged in and selected the data hubs, they are first shown the search web-page, illustrated in Figure 2. The dashboard search page allows the user to uniformly explore the data streams contained in each hub.

The Hypercat search mechanisms described in section II-B are made available on this web-page with a tab for each search type. Search mechanisms currently supported are: a) tag names, b) data types (relation-value pairs, including lexical searching), and c) geographical areas. While browsing the search results, as illustrated in Figure 3, users can check that potentially useful streams are functioning properly by displaying a chart of the last 12-24hrs results. To select a stream for manipulation and/or analysis, the user provides their own unique stream name and then clicks on the button with a lightning bolt symbol which will make it *live* and add it to the *database streams* list, viewed in a separate tab (Figure 4). All streams in this list are either *live*, *previous* or *derived*. Each item in the *Database Streams* list also includes a check-box, so that the user can select streams as input to the following 3 options, (as shown in Figure 4): (i) data manipulation; (ii) data analysis; and (iii) data download (to save to a local file).



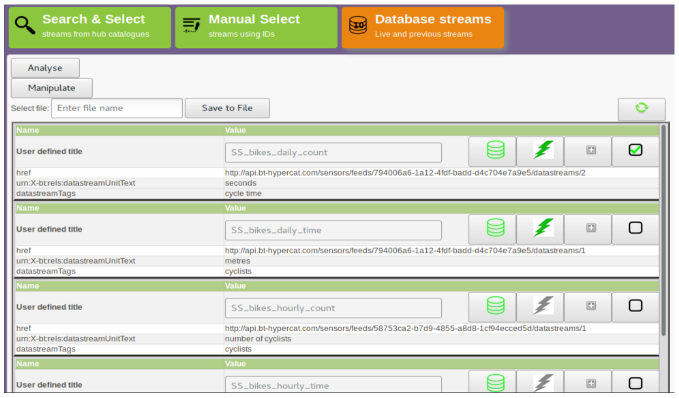
Select results to be added to live streams

| Name | Value |
|---|---|
| urn:X-bt:rels:feedTitle | Capacity |
| feedTags | Triangulum, Transport & Mobility, CarParks, Aquatic Centre, Capacity |

| Name | Value |
|---|---|
| User defined title | |
| href | https://130.88.97.137/piwebapi/attributes/A0EXpbRmwnc7kq0OSy1LydJJQ-aUqjTuZ5xGA7ABQVqtCQgagMTtskezVwGiTW04dnqxwVk0tUEktUDAxLkRTLk1BTi5BQy5VS1xUUklBTkdVTFVNXFRSQU5TUE9SVCAmlE1PQklMSVRZXENBUlBBUktTXEFRVUFUSUMgQ0VOVFJFfEZSRUUgU1BBQ0VT |
| urn:X-hypercat:rels:hasDescription:en | |
| urn:X-bt:rels:feedTitle | Aquatic Centre |
| urn:X-hypercat:rels:isContentType | application/json |
| http://www.w3.org/2003/01/geo/wgs84_pos#lat | 53.4687497271 |
| http://www.w3.org/2003/01/geo/wgs84_pos#long | -2.23231995128 |
| feedTags | Triangulum, Transport & Mobility, CarParks, Aquatic Centre |
| urn:X-hypercat:rels:hasDescription:en | Description: Number of free spaces available in the car-park. Source: Transport for Greater... |
| urn:X-bt:rels:datastreamUnitText | Free Spaces |
| urn:X-bt:rels:feedTitle | Free Spaces |
| urn:X-hypercat:rels:isContentType | application/json |
| urn:X-bt:rels:belongsToSensorFeed | https://130.88.97.137/piwebapi/elements/E0XpbRmwnc7kq0OSy1LydJJQ-aUqjTuZ5xGA7ABQVqtCQgVk0tUEktUDA... |
| http://www.w3.org/1999/02/22-rdf-syntax-ns#type | triangulum#attribute |
| feedTags | Triangulum, Transport & Mobility, CarParks, Aquatic Centre, Free Spaces |

| Name | Value |
|---|---|
| User defined title | AQ_Occ |
| | https://130.88.97.137/piwebapi/attributes/A0EXpbRmwnc7kq0OSy1LydJJQ- |

Fig. 3. Search/Fetch page: search results list with expanded stream item



| Name | Value |
|---|---|
| User defined title | SS_bikes_daily_count |
| href | http://api.bt-hypercat.com/sensors/feeds/794006a6-1a12-4fdf-badd-d4c704e7a9e5/datastreams/2 |
| urn:X-bt:rels:datastreamUnitText | seconds |
| datastreamTags | cycle time |

| Name | Value |
|---|---|
| User defined title | SS_bikes_daily_time |
| href | http://api.bt-hypercat.com/sensors/feeds/794006a6-1a12-4fdf-badd-d4c704e7a9e5/datastreams/1 |
| urn:X-bt:rels:datastreamUnitText | metres |
| datastreamTags | cyclists |

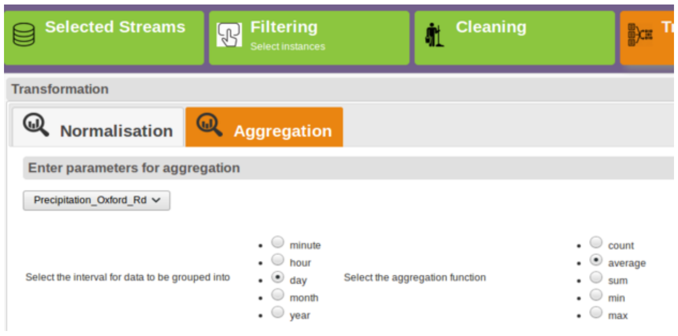
| Name | Value |
|---|---|
| User defined title | SS_bikes_hourly_count |
| href | http://api.bt-hypercat.com/sensors/feeds/58753ca2-b7d9-4855-a8d8-1cf94ecced5d/datastreams/1 |
| urn:X-bt:rels:datastreamUnitText | number of cyclists |
| datastreamTags | cyclists |

| Name | Value |
|---|---|
| User defined title | SS_bikes_hourly_time |

Fig. 4. Search/Fetch page: database streams list (no items expanded)

## C. Data manipulation

Sensor data manifests features that make data analysis more challenging, such as missing data, noise, and the level of data heterogeneity. When visualising and/or integrating multiple data streams, the user needs to be able, for example, to replace missing data points, or to make different data sets more comparable using techniques such as normalisation. Feature subset selection might also be used to remove irrelevant



Enter parameters for aggregation

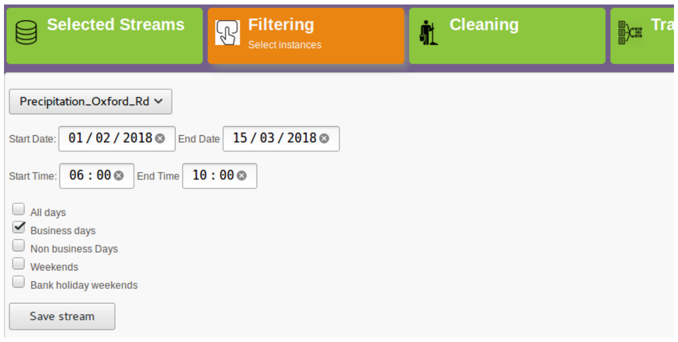
Precipitation_Oxford_Rd

Select the interval for data to be grouped into: minute, hour, [✓ day], month, year

Select the aggregation function: count, [✓ average], sum, min, max

Fig. 5. Data manipulation page: aggregation sub-tab (in transformation tab)

and redundant features or to reduce data dimensionality; the dashboard is designed to eventually cover each of the main data pre-processing steps in [27]. Once the user has selected streams and clicked the *Manipulate* button on the *Database streams* tab (Figure 4), a new *data manipulation* web page opens (Figure 5). This contains a *Selected Streams* tab which displays the selected stream(s) data as recorded in InfluxDB; a *Filtering* tab for selecting instances (Figure 6) and a further four tabs for each type of data manipulation. Each tab contains a sub-tab for each available pre-processing technique and these contain the available options. Once the selected techniques and parameters have been input, the user clicks the *save* button, the manipulation query parameters are sent to the *query processor*, continuously run against the respective streams and saved as *derived* streams in InfluxDB. The *query processor* uses InfluxDB's inbuilt CQ mechanism where possible, and supplements this with further query functionality where pre-processing techniques [27] are not implemented.



Precipitation_Oxford_Rd

Start Date: 01 / 02 / 2018 End Date 15 / 03 / 2018

Start Time: 06 : 00 End Time 10 : 00

- ☐ All days
- ☑ Business days
- ☐ Non business Days
- ☐ Weekends
- ☐ Bank holiday weekends

Fig. 6. Data manipulation page: filtering tab

The manipulation techniques currently available are:

1) *Filtering*: the selection of instances using criteria such as dates, types of day (bank holidays for example) (Figure 6) and geographical location ;
2) *Cleaning*:
    a) *Missing values*: either ignoring them or calculating the mean of existing values to use as substitutes;
    b) *Outlier smoothing*: allows binning/clustering methods to detect and find replacements for outliers.
3) *Transformation*:

a) *Normalisation*: adjusting the scale of a stream's value range, to match the ranges of others;

b) *Aggregation*: abstracts to a higher level view of the data to improve analysis results. This is done by grouping the values of multiple datapoints into a single, more useful value, for example grouping a car park's *used spaces* values into hourly groups and taking the mean of each group.

4) *Discretisation*: partitions values that lie on a continuous scale, into nominal attributes or intervals; e.g., converting temperature values into low, medium and high.

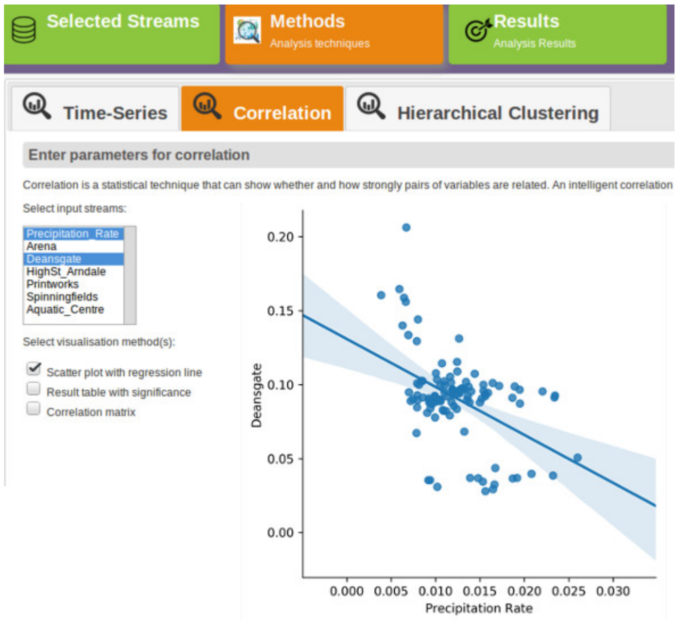
Streams may be subject to various state changes along a pre-processing pathway and to allow this, the *derived* streams held in InfluxDB are made available in the *database streams* list (Figure 4) for further manipulation (or deletion).

### D. Analysing data streams

To show how users can study and explore the streams to see whether the data contains interesting patterns, a set of typical analysis and visualisation methods have been implemented. The dashboard allows users to analyse individual streams and/or multiple streams and combining both options allows the users to view and compare several streams. The analysis and visualisation techniques currently available are:

1) *Time series analysis*: The dashboard allows the user to conduct simple time series analysis, useful for modeling the observations over time to look for trends, seasonality and other cyclic patterns and allowing the prediction of future values. These observed patterns might be the end goal for the analyst, or a step in a higher level methodology (such as the *Box-Jenkins* methodology [31]) where these patterns need to be accounted for, before finding further models to explain any remaining *randomness* in the data. The dashboard allows the observation of *several* time series at once.
*Visualisations*: The line graph and scatter plots are simple ways to display time series data, with the y-axis value representing each data value and the x-axis representing time.

2) *Correlation analysis*: allows comparison of pairs of variables to search for the strength of relationship between the two (for example between precipitation and car park use). Discovering how one stream correlates with another can give users an idea of certain associations. Although this technique only allows comparison of *two streams* at once, if correlation matrix visualisation is used, a set of correlation results between a list of stream pairs can be viewed and compared.
*Visualisations*: Correlation results between a single pair of streams are displayed in a table with the correlation values. Scatter plots are also used in the dashboard, with an axis for each variable and a *least squares regression line* showing the most likely correlation pattern by drawing the straight line with the least distance from each of the datapoints in the plot (as shown in Figure



Enter parameters for correlation

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related. An intelligent correlation

Select input streams:

Precipitation_Rate
Arena
Deansgate
HighSt_Arndale
Printworks
Spinningfields
Aquatic_Centre

Select visualisation method(s):

[✓] Scatter plot with regression line
[ ] Result table with significance
[ ] Correlation matrix

Fig. 7. Data analysis page: scatter plot of rainfall and a car park

7). The correlation matrix (Figure 8) uses a heat map display to allow the user to visualise many correlation results at one time.

3) *Hierarchical clustering*: This method involves the dividing of instances into a hierarchy of clusters based on a set of features. In the hierarchy, each node represents a cluster of the datapoints. Hierarchical clustering does not require the number of clusters as an input, but a termination condition should be defined to identify when the merge process should be finished. Applying clustering on time-series data, where the clustering is based on stream features such as time-periods and/or locations, has the potential to identify similar patterns occurring at multiple times [32]. This technique allows observation of *two or more streams* at once.
*Visualisations*: To visualise the resulting clusters we display a dendrogram with heat map, in which a colour bar is provided to represent the range of values assigned to each cluster.

To perform data analysis, the user selects the desired streams from the *database streams* tab and clicks the *Analyse* button (Figure 4). A *data analysis* web page opens (Figure 7), containing a *Selected Streams* tab which displays the selected stream(s) data as recorded in InfluxDB and a *Methods* tab which contains sub-tabs for the available analysis methods.

## IV. Case Studies

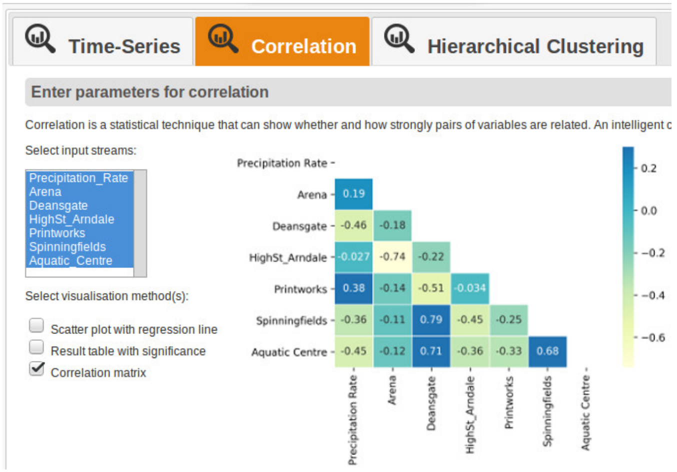
### A. Precipitation and car park use

*Does the early morning rainfall affect commuters decision to drive to work?* Our first case study illustrates the search and fetch process by demonstrating the search for streams relating to city centre car parks and rainfall. To obtain these the user specifies tag names, such as *car parks*, *weather* and *precipitation rate*, along with a geographical area covering the city centre (Figure 2). When the *Results* tab is clicked, the application searches for and returns matching items on that tab (Figure 3), in this case a selection of city centre car park and weather streams. The user browses this list and for each stream that they wish to analyse, they give it a unique name and then click on the button with the *lightening bolt* symbol for that stream, which adds it to the *Data Streams* list (Figure 4) and puts it into the *live* state (see III-A). The longer the web-page is left open, with the streams in the *live* state, the more data is collected from hubs and stored in InfluxDB.

To pre-process the datastreams, the user firstly browses the *Database Streams* list and selects the *precipitation* stream. They click on the *Manipulate* button to open a *Data Manipulation* web-page for this stream and select the required pre-processing steps and parameters. The user clicks the *save* button and the required pre-processing methods are applied to that stream and it is saved as a *derived* stream. This is repeated for each *car park free spaces* data sets. The pre-processing steps required for this case study are:

1) *data cleaning*: in the case study, the required cleaning step is to check for missing values. To do this, the user can browse all datapoints stored in InfluxDB by clicking on the *Selected Streams* sub-tab shown in Figure 5, and assess whether there is a need to process them. In this case they decide to ignore all missing values and select that option on the *Missing Values* sub-tab.
2) *filtering*: as the hypothesis relates to citizens' decisions to drive into the city during the mornings, the user needs to select only rainfall data instances collected during weekday mornings, between 6am and 10am. In addition the user is only interested in data for working days, so both precipitation and car park streams are filtered to select only business/working days. The user selects the *Filtering* tab (Figure 6) to perform these selections.
3) *data reduction* The required data reduction steps are aggregation (Figure 5) to obtain the average rainfall for each day and the minimum car-parking spaces that were available per day, per car park. Generalisation is then used to categorise the values into low, medium and high.

In this case study, the user would like to compare the pre-processed rainfall and car-park spaces using scatter-plot visualisations and run further data analysis using correlation tables. In Figure 4 they have selected their newly created *derived* streams and then clicked on the *Analysis* button, opening the corresponding web-page. In Figure 7 our user has selected *Scatter*. One of the pre-processed car park streams and the preprocessed precipitation stream (both *derived*) are selected and the *scatter plot with regression line* option is checked. The result shows that an inverse correlation exists between these two streams: when precipitation is high, the number of free car park spaces is lower, with a correlation of -0.46; this moderate correlation might be deemed worthy of further investigation. The user might then decide to look at correlations between all variables, so they would check the *correlation matrix* box; the



Enter parameters for correlation

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related. An intelligent c

Select input streams:

Precipitation_Rate
Arena
Deansgate
HighSt_Arndale
Printworks
Spinningfields
Aquatic_Centre

Select visualisation method(s):

- [ ] Scatter plot with regression line
- [ ] Result table with significance
- [x] Correlation matrix

| | Precipitation Rate | Arena | Deansgate | HighSt_Arndale | Printworks | Spinningfields | Aquatic Centre |
|---|---|---|---|---|---|---|---|
| Precipitation Rate | | | | | | | |
| Arena | 0.19 | | | | | | |
| Deansgate | -0.46 | -0.18 | | | | | |
| HighSt_Arndale | -0.027 | -0.74 | -0.22 | | | | |
| Printworks | 0.38 | -0.14 | -0.51 | -0.034 | | | |
| Spinningfields | -0.36 | -0.11 | 0.79 | -0.45 | -0.25 | | |
| Aquatic Centre | -0.45 | -0.12 | 0.71 | -0.36 | -0.33 | 0.68 | |

Fig. 8. Data analysis page: correlations sub-tab

output from running this analysis is illustrated in Figure 8.

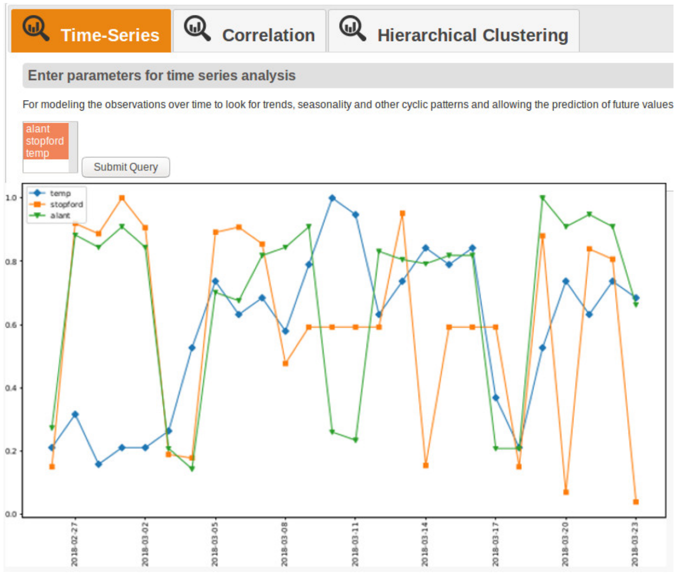
## B. Weather and building energy consumption

*Does weather affect the energy consumption of University of Manchester buildings?* In this case study, the user wishes to check the proportional changes in energy consumption in University of Manchester buildings as the external temperature changes. As in the previous case study, the *search and fetch* web page is used to search for relevant streams, in this case relating to buildings energy data and temperature. To obtain these the user specifies tag names, such as *building*, *energy*, *University of Manchester* and *temperature*. The user decides to compare the *CO2 electricity* consumption of an old and a new building, and they choose the Stopford building (1972) and the newer Alan Turing building (2007). These are added to the *Data Streams* list along with a temperature stream (Figure 4), therefore placing each one into the *live* state.

The user selects the required streams from the *Database Streams* list and clicks on the *Manipulate* button to open a *Data Manipulation* web-page for these streams and select the required pre-processing steps and parameters.

The pre-processing steps required for this case study are:

1) *data cleaning*: the Stopford Buildng electricity consumption stream is found to have a large proportion of missing values, so the *Cleaning* and then *Missing Values* tabs are selected and *Use mean* option is selected, which results in the filling in of the missing values using the mean value of the dataset for each 24 hours.
2) *transformation* The required transformation steps are normalisation and aggregation (Figure 5). As the value ranges in all three selected streams differ greatly from one another, and the user is only interested in proportional changes, the values of each dataset are normalised to lie between zero and one. The streams are also aggregated to contain only the maximum value per day, thus showing a more generalised view of the data.


Time-Series Correlation Hierarchical Clustering

Enter parameters for time series analysis

For modeling the observations over time to look for trends, seasonality and other cyclic patterns and allowing the prediction of future values

alant
stopford
temp

Submit Query

Fig. 9. Data analysis page: time series sub-tab

Again, the user clicks the *save* button, and the required pre-processing methods are applied to these streams and they are saved as a *derived* streams.

The user would like to run a time series analysis/visualisation of this data. They select the relevant *derived* streams and then clicked on the *Analysis* button, opening the corresponding web-page. In Figure 9 our user has selected the *Time Series* sub-tab of the *Method* tab. All three streams are selected. The resulting visualisation gives the user a general idea of how the energy consumption changes as the temperature changes.

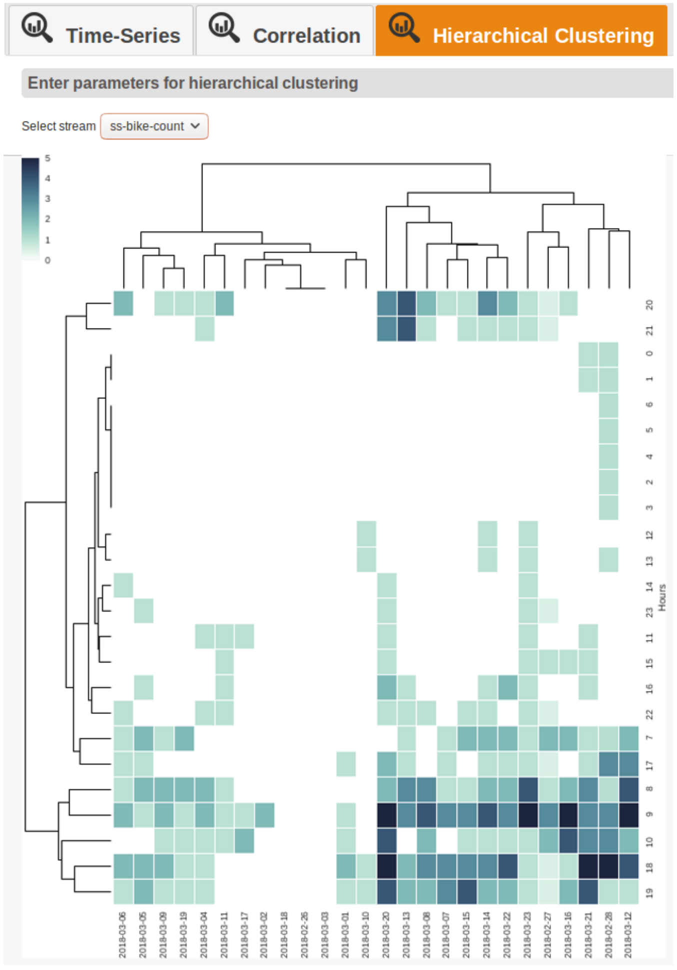
### C. Bicycle trial results: show usage at different days and times

*Use See.Senses bicycle data to obtain an overview of bicycle usage at different times of day.* The user wishes to view live crowd-sourced data from a bicycle usage trial run by BT and See.Sense. The trial employs 180 volunteer cyclists who track their movements around the city centre using a smart phone application and the *cyclist-count* per hour data-stream can be used to measure the level of cycle *usage*.

Once the user has searched for See.Sense cycling data streams, they select the required *cyclist-count* stream from the *Database Streams* list, click on the *Manipulate* button to open a *Data Manipulation* web-page for this stream, and select the required pre-processing steps and parameters.

The only pre-processing step required for this case study is *data cleaning*. From observing the raw data in the *Database Streams* list, it can be seen that very few values are missing, and as the user only requires a general overview of the data, they choose to ignore these values. Again, the user clicks the *save* button, and the required pre-processing method is applied to these streams and the result is saved as a *derived* stream.

To obtain an overview of cycling patterns by day and hour, the user runs a hierarchical clustering analysis of this data. For this case study the method involves clustering based on the count of cyclists per hour. The y-axis represents clustering


Time-Series Correlation Hierarchical Clustering

Enter parameters for hierarchical clustering

Select stream ss-bike-count

Fig. 10. Data analysis page: hierarchical clustering sub-tab

of hours of the day (0-23) and the x-axis displays the clustering based on the included days (26 days from 26/02/2018 to 23/03/2018). The user is able to identify similar patterns occurring at multiple times, and the resulting dendrogram with heatmap shows the expected patterns for hours of the day, with the *rush-hour* times from 7 to 10 and from 17 to 19 having the highest counts. This visualisation also shows that cycling usage is extremely low on several days, and upon further checking these are found to coincide with known periods of freezing temperatures and snowfall. It can also be seen that although counts are generally fewer during weekends, the hourly patterns do not differ much.

## V. Conclusion

We present a smart city dashboard to allow interactive discovery, transformation, integration and analysis of data. This enables stakeholders (town planners, data scientists and those who live and work in the city) to obtain insights of potential utility with modest levels of effort. We illustrate this by three case studies presenting user interaction with the dashboard to test hypotheses such as does *rainfall affects people's decision to drive into the city centre on work days*. These show how the system brings together the above techniques to establish whether correlations exist between data sets from multiple themes. The CityVerve data dashboard allows greater flexibility than typical city dashboards, as the user can explore all published city datastreams on the selected hubs and choose from a combination of pre-processing and analysis methods to manipulate/visualise the datasets. In contrast, existing smart city data dashboards are designed to allow the user to browse a preset list of data sources, built in to the dashboard, to inform specific city tasks such as garbage collection or local authority spending. Similarly, the analysis methods used in these applications are hard-wired and bespoke, presenting results and visualisations to the user to answer specific, predefined questions. Whilst custom-made city dashboards provide a valuable means to allow non-technical *domain experts* to navigate the complex *smart city* information space, we present a more flexible approach to allow both easy exploration of city data streams and further experimentation with a variety of pre-processing and analysis methods.

In future work, the dashboard will be advanced to accommodate data manipulation tasks such as advanced data discretisation methods; data analysis methods will be expanded to include further data analytic tasks shown to be useful for time-series data such as classification, rule discovery and novelty detection. In addition, to make the research reproducible, the software will soon be made publicly available (http://doi.org/10.5281/zenodo.1253279).

## Acknowledgment

The authors acknowledge support from the Innovate UK CityVerve project 102561; we also wish to thank all colleagues involved in the project consortium.

## References

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[2] E. Al Nuaimi, H. Al Neyadi, N. Mohamed, and J. Al-Jaroodi, "Applications of big data to smart cities." *Journal of Internet Services and Applications*, vol. 6, no. 1, p. 25, 2015.

[3] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[4] S. S. Husain, A. Kalinin, A. Truong, and I. D. Dinov, "Socr data dashboard: an integrated big data archive mashing medicare, labor, census and econometric information," *Journal of big data*, vol. 2, no. 1, p. 13, 2015.

[5] E. Zhu, K. Q. Pu, F. Nargesian, and R. J. Miller, "Interactive navigation of open data linkages," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1837–1840, 2017.

[6] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, "Urban planning and building smart cities based on the internet of things using big data analytics," *Computer Networks*, vol. 101, pp. 63–80, 2016.

[7] T. Jaffey, J. Davies, and P. Beart, "Hypercat 3.00 specification," *Hypercat Limited*, 2016.

[8] M. d'Aquin, J. Davies, and E. Motta, "Smart cities' data: Challenges and opportunities for semantic technologies," *IEEE Internet Computing*, vol. 19, no. 6, pp. 66–70, 2015.

[9] G. Pan, G. Qi, W. Zhang, S. Li, Z. Wu, and L. T. Yang, "Trace analysis and mining for smart cities: issues, methods, and applications," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 120–126, 2013.

[10] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma, "The role of big data in smart city," *International Journal of Information Management*, vol. 36, no. 5, pp. 748–758, 2016.

[11] A. Whitmore, A. Agarwal, and L. Da Xu, "The internet of things? a survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[12] J. A. G. Macias, J. Alvarez-Lozano, P. Estrada, and E. A. Lopez, "Browsing the internet of things with sentient visors," *Computer*, vol. 44, no. 5, pp. 46–52, 2011.

[13] R. J. Crouser, L. Franklin, and K. Cook, "Rethinking visual analytics for streaming data applications," *IEEE Internet Computing*, vol. 21, no. 4, pp. 72–76, 2017.

[14] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. Pardo, and H. Scholl, "Understanding smart cities: An integrative framework," in *System Science (HICSS), 45th Hawaii I. Conference on*, 2012, pp. 2289–2297.

[15] A. Zanella, N. Bui, L. V. A. Castellani, and M. Zorzi., "Internet of things for smart cities," *Internet of Things*, vol. 1, no. 1, pp. 22–32, 2014.

[16] R. Lea, "An overview of the technology trends driving smart cities." Tech. Rep., 2015.

[17] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. In Press, 2017.

[18] H. Pineo, K. Glonti, H. Rutter, N. Zimmermann, P. Wilkinson, and M. Davies, "Characteristics and use of urban health indicator tools by municipal built environment policy and decision-makers: a systematic review protocol," *Systematic reviews*, vol. 6, no. 1, p. 2, 2017.

[19] S. Few, *Information Dashboard Design - The effective visual communication of data.* O'Reilly, 2006.

[20] M. Mendonça, B. Moreira, J. Coelho, N. Cacho, F. Lopes, E. Cavalcante, A. Dias, J. L. Ribeiro, E. Loiola, D. Estaregue *et al.*, "Improving public safety at fingertips: A smart city experience," in *Smart Cities Conference (ISC2), 2016 IEEE International.* IEEE, 2016, pp. 1–6.

[21] G. McArdle and R. Kitchin, "The dublin dashboard: Design and development of a real-time analytical urban dashboard," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, p. 19, 2016.

[22] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, and U. Srivastava, "Stream: The stanford data stream management system," *Data Stream Management. Data-Centric Systems and Applications*, 2016.

[23] U. Cetintemel, J. Du, T. Kraska, S. Madden, D. Maier, J. Meehan, A. Pavlo, M. Stonebraker, E. Sutherland, N. Tatbul *et al.*, "S-store: a streaming newsql system for big velocity applications," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1633–1636, 2014.

[24] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management," *the VLDB Journal*, vol. 12, no. 2, pp. 120–139, 2003.

[25] A. Elmore, J. Duggan, M. Stonebraker, M. Balazinska, U. Cetintemel, V. Gadepally, J. Heer, B. Howe, J. Kepner, T. Kraska *et al.*, "A demonstration of the bigdawg polystore system," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1908–1911, 2015.

[26] S. Mallapuram, N. Ngwum, F. Yuan, C. Lu, and W. Yu, "Smart city: The state of the art, datasets, and evaluation platforms," in *Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on.* IEEE, 2017, pp. 447–452.

[27] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data preprocessing for supervised learning," *International J. of Computer Science*, vol. 1, no. 2, pp. 111–117, 2006.

[28] P. Zikopoulos, C. Eaton *et al.*, *Understanding big data: Analytics for enterprise class hadoop and streaming data.* McGraw-Hill Osborne Media, 2011.

[29] F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong, "Data mining for the internet of things: Literature review and challenges." *International Journal of Distributed Sensor Networks*, vol. 2015, 2015.

[30] R. Lea, "Hypercat: an iot interoperability specification," 2013.

[31] G. E. Box and M. Gwilym, "Jenkins. time series analysis forecasting and control," *São Francisco: Holden-Day*, 1970.

[32] S. Aghabozorgi, A. Shirkhorshidi, and T. Wah, "Time-series clustering - a decade review," in *Information Systems*, vol. 53, 2015, pp. 16–38.