An Autonomic Approach to Real-Time Predictive Analytics using Open Data and Internet of Things

Wassim Derguech, Eanna Bruke, Edward Curry
Insight Centre For Data Analytics - National University of Ireland, Galway
wassim.derguech@insight-centre.org, eannaburke1@gmail.com, ed.curry@insight-centre.org

***Abstract*—Public datasets are becoming more and more available for organizations. Both public and private data can be used to drive innovations and new solutions to various problems. The Internet of Things (IoT) and Open Data are particularly promising in real time predictive data analytics for effective decision support. The main challenge in this context is the dynamic selection of open data and IoT sources to support predictive analytics. This issue is widely discussed in various domains including economics, market analysis, energy usage, etc. Our case study is the prediction of energy usage of a building using open data and IoT. We propose a two-step solution: (1) data management: collection, filtering and warehousing and (2) data analytics: source selection and prediction. This work has been evaluated in real settings using IoT sensors and open weather data.**

Keywords*—Predictive Analytics; Autonomic System; Machine Learning; IoT; Open Data; Energy Management;

I. Introduction

A relatively recent development in the field of computer science is the concept of the Internet of Things (IoT), a vision where the planet is covered with sensors, from wireless sensors monitoring the welfare of farm animals to Internet connected utilities, traffic monitors, and many other devices. Today there are approximately 9 billion devices connected to the Internet with this number set to reach 15 billion by 2015 [1]. IoT is enabling a future of smart environments, e.g. smart cities, smart buildings and smart grids.

In 2011, IBM estimated that 90% of data created by mankind had been created in the previous 2 years [2]. Data being generated by a multitude of Internet connected devices can quickly become infeasible to cope with in traditional ways. At the same time, autonomic computing systems are being developed as a way to cope with large and increasingly complex systems. Autonomic systems can manage themselves when given high-level objectives from administrators, freeing them from low-level tasks [3]. If an infrastructure is to cope with Big Data, it needs to be self-managing to remove as much of the burden as possible.

This paper aims to provide a methodology for managing Open Data collected from the web or from IoT sensors for effective decision making in the context of a smart environment. A smart environment contains embedded sensors and actuators to make the environment as a whole more efficient, more reliable and more comfortable for its occupants [4]. A smart building is a localisation of this idea and addresses a subset of the needs of a smart environment, e.g. adjusting building systems in response to occupant comfort and energy demands.

In Predictive Analytics, knowledge of an event occurring before it occurs enables action to be taken to mitigate the occurrence of the event or prepare for its eventuality. The concept of Predictive Analytics applied to building energy management can provide insight into the future energy usage; this can enable scheduling of building operations in a more efficient and effective manner. This paper is concerned with the evaluation and selection of optimal sensor data sources from the Internet of Things that are most suited to prediction of power use in a building. The Web source selection process is demonstrated in an autonomic building management environment where it is used to select weather data upon which predictions for future electricity usage are made.

In the same context, the literature proposes various solutions [5], [6], [7], [8], [9] for predicting energy use using weather data. Unlike most of these contributions that use a single source of weather data, we propose in our work a methodology that uses multiple sources. Additionally, we focus on proposing an *autonomous system* [10] that provides accurate results rapidly even without historical data.

The remainder of the paper is organized as follows: After defining the context and motivating the research problem in Section II, Section III details our proposed multi-layer open data management architecture. Then in Section IV, we discuss our approach for the selection of data from the Internet and sensor networks for effective predictive analytics. Section V lists our technology choices for the development of the proposed system. For evaluating the autonomous aspect of the designed system as well as the used machine learning algorithms, we carried out three experiments reported in Section VI. Before concluding the paper in Section VIII, we analyse related works in Section VII.

II. Context and Motivation

The aim of this work is to use open weather data accessible from free APIs provided by various online sources in combination with building electricity use data from local sensors to predict future electricity use in an accurate and reliable manner. When working with data taken from sensors connected to the Internet and other Web sources, the data is from a 3rd party and the quality and reliability is outside of our control. This leads to the problem of having to pick data sources from the Web that best suit our needs. In addition, this must be done in such a way that the system is reliable and continues making the best possible predictions over time so that prediction consumers can depend on the quality of the predictions made.

This work attempts to develop a way of quickly and efficiently evaluating open weather data sources to be used in predictive analytics. The source selection is part of an autonomic system that maintains and improves the quality of the predictions over time while being self-managing [10]:

- Self-configuration:
  - Automatic installation and initiation: The system can be installed into any building and automatically start being useful with minimal intervention by a skilled worker.
  - Easily generalised to any building: There should be low configuration effort to adapt the system to another building to encourage re-use.
- Self-optimization:
  - Select the best data sources from the Web: The system chooses the best sources of open weather data to make the best possible predictions for future power consumption.
  - Adapt to changes in the building use: The predictions should react to changes in building use, e.g. expansion or contraction of a workforce, extensions, renovations etc.
  - Low user interaction: The system should continue working with no supervision.
- Self-healing:
  - Transparency to end users in the case of failure of a data source: In the case of a failure of a data source (i.e. a weather station malfunction), the system should continue to make a best effort prediction so that agents dependent on it can continue to operate.
  - Continuously maintain good predictions: Predictions must remain accurate as poor predictions may cause consumers of the data to make wrong decisions.
  - Quick identification of faults: Faulty data sources (e.g. a damaged sensor) should be identified quickly so that an alternative data source can be used.

In order to build an efficient energy prediction systems in such context, two main challenges need to be handled: *heterogeneous data management* and *source selection*.

1) Heterogeneous data management: In smart environments, sharing information and data is a big challenge due to various aspect. Indeed, data produced in such environments is dynamically and heterogeneously generated by different applications in different domains across different enterprises, etc. Consequently, there is a need to transform data in a format that is easily exchangeable and integrateable. A promising solution in this context is the use of Linked Data.

Linked data is set of best practices for representing information in RDF format and relating or connecting this information. The basic ingredient of linked data is structured data and links between structured data. The main philosophy of linked data is to create data that can be shared and reused.

2) Data Source Selection: The selection of data sources is important as the data from the sources influence the results of predictive analytics. In a 2011 review of trust in networked data sets [11], authors note that the process of selecting a data source is subjective based on the needs of the consumer. A common method for selecting a dataset to answer a query is to examine the metadata associated with the data source, e.g. size of the dataset, date and frequency of updates [12]. Another method for determining correct information is to establish a consensus from several sources [11]. Our proposed methodology for selecting the right data source consists of evaluating the results of the predictive analytics and select the data source with the best results.

III. Open Data Management

For predictive analytics, data needs to be collected, converted and stored in a proper format. We propose in this section a multi-layer architecture for managing open data. After an overview of this architecture in Section III-A, we discuss data linking formalism used in this work in Section III-B.

A. Architecture Overview

We consider three primary types of open data: i.e., weather observation, weather forecast and IoT sensor data. We defined a multi-layer data management architecture that performs data collection, conversion and storage as illustrated in Fig. 1.

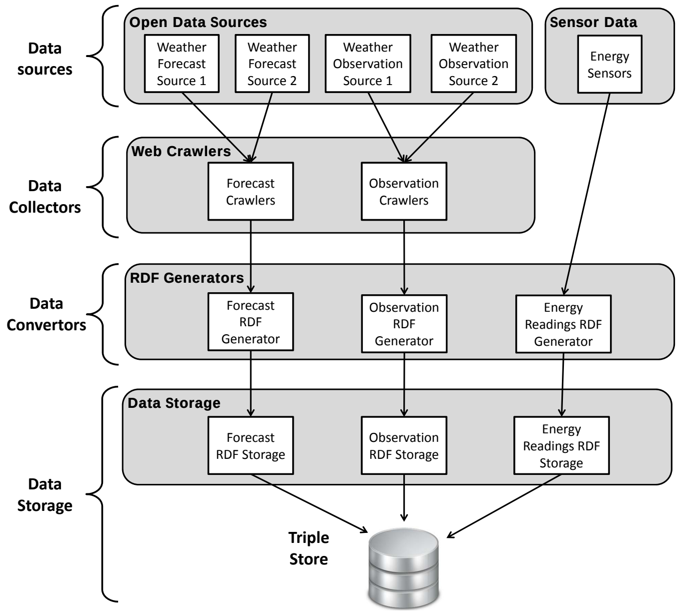


Fig. 1. Data Management Architecture

1) Data Sources layer: The top layer of the proposed architecture introduces the sources of data to be considered in the entire system. We consider at least two sources of weather observation and weather forecast (open data) and at least one source of energy reading (sensor data). There is no restriction on these sources, they can be a combination of Web services and devices from the Internet of Things.

2) Data Collectors layer: For each data source, a dedicated data collector is designed. The data is collected in its individual format and access mechanism, e.g., HTTP requests or download from FTP servers, then sent to the converter.

3) Data Converters layer: The objective here is to receive messages from data collectors and convert them to RDF using predefined ontologies for every data type. Ontologies are discussed in Section III-B.

4) Data Storage layer: Listeners are created for each type of RDF which persist the data into the triple store.

B. Linked Data Ontologies

Ontologies constitute formal specifications of shared conceptualisations [13] which, in essence, fosters the reuse of existing assets. To this end, the available ontologies relating to sensor readings and weather data should be surveyed before a choice is made whether to use an existing ontology, extend an existing or develop a new one.

We reviewed a set of proposed ontologies with respect to these criteria:

- *In use*: The ontology should be in use and well documented for easy learning.
- *Relevance*: The ontology should correspond to the conditions experienced by the building. Ontologies should correspond to surface readings and not for describing space or marine meteorological conditions.
- *Numerical*: The ontology needs to be numerical and precise to be suitable for machine learning. Vague terms such as "Hot" or "Humid" would lead to interpretation problems and imprecision.

1) IoT Sensor Readings Ontology: We propose the use of Semantic Sensor Network Ontology (SSN for short) [14] that is developed by W3C Semantic Sensor Networks Incubator Group (SSN-XG) for describing sensors and related concepts.

2) Weather Observation Ontology: We recommend using AEMET Weather Observation Ontology [15] for describing weather observations. This ontology was developed by the Ontology Engineering Group at Universidad Politécnica de Madrid and has been been proven to be effective in large deployment. Details of its creation and implementation are available in [16].

3) Weather Forecast Ontology: The recommended ontology is the "meteo" ontology developed by Sean B. Palmer, as it was the only ontology that satisfies the three criteria [17]. The ontology is very simple, listing times a prediction was made, when the prediction is for and the expected values of the meteorological phenomena at each time. This ontology also aligns well with the weather observation ontology from the Ontology Engineering Group.

IV. Source Selection and Predictive Analytics

This section discusses the current approach taken for the selection of data sources from the Internet and sensor networks.

The prediction component is designed according to the architecture depicted in Fig. 2. It consists of five components: i.e., Error Comparison module, Re-selection Controller, Source Selector, error database and the user interface. The database holds the prediction results over time. It is queried at intervals by the re-selection controller to determine how well the model

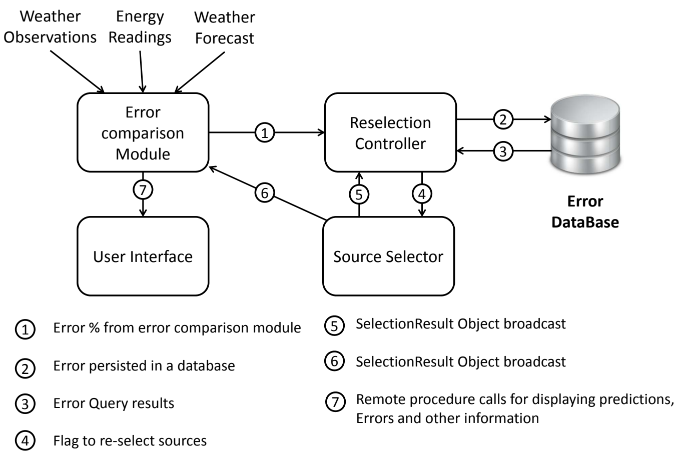

Weather Observations
Energy Readings
Weather Forecast
Error comparison Module
Reselection Controller
Error DataBase
User Interface
Source Selector
1 Error % from error comparison module
2 Error persisted in a database
3 Error Query results
4 Flag to re-select sources
5 SelectionResult Object broadcast
6 SelectionResult Object broadcast
7 Remote procedure calls for displaying predictions, Errors and other information

Fig. 2. Source Selection and Predictive Analytics Architecture

is performing. The user interface is the actual consumer of the predictions made by the system. It informs users how the system is performing, showing information about the weather sources being used, predictions for the near future, the current power use, and the historical error of the system. As we propose to design an autonomic system, there is no opportunity or necessity for human input through the interface.

A. Error Comparison Module

The error comparison module makes predictions of power consumption which can be queried by any interested parties (e.g., the UI). The predictions are held for later comparison with the actual power use. Predictions are also made using the current conditions and compared with the current power readings as an indication of the current performance of the system and to identify when the power use is unusually high and investigate if actions need to be taken. Additionally, predictions are compared with actual power readings to generate an error percentage, which is then displayed in the UI and sent to the Re-Selection controller for analysis and storage.

Please note that we use the Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) as error indicators.

B. Re-selection controller

The re-selection controller is responsible for controlling when to select the sources used for prediction. Re-selects involve building many prediction models, which is costly and time consuming, so trade-offs should be made between the frequency of re-selections and maintaining the best prediction model possible. We suggest to use hard limits of 15 minutes and one month for the upper and lower bounds of the re-select interval, but the exact value interval should be kept dynamic. The criteria for re-selections are shown in Table I and discussed further below.

1) Timed error check: is done by checking the recent performance against the expected performance of the prediction model. This check is for long term trends in the data set. Error checks are less costly than builds so they happen on a more frequent basis than timed builds. The error check becomes

TABLE I. SOURCE RE-SELECTION CRITERIA

Reasons to re select the sources used	**Reasoning**	**How performed**	**Autonomic characteristic addressed**
Timed error checks	Checking if the prediction model has become less accurate.	Query the internal error DB for the sources being used.	*Self-optimising* *Self-healing*
Timed builds (regardless of errors)	Data collected may allow a more accurate prediction model than is currently being used.	Send flag to re-select	*Self-optimising*
Very high error in the incoming stream	May Indicate a failure in a sensor or data source.	Short-term Error check	*Self-healing*
New source event received	New source may be more accurate than existing sources.	Send flag to re-select	*Self-configuring* *Self-optimising*

more frequent every time the error is returned is too high and less frequent when the error found is under the acceptable threshold. Changeable conditions make it check more often to include the newest and most relevant data and prevent data aging. The error check is for 25% of the time the prediction model is in place, so that the newest data is given high priority.

The error check uses Student's t-distribution to determine when the mean error is too high. It does this by checking that the average error is lower than a threshold computed using: $Threshold$ = ($expectedMAPE$) + 0.674 * ($standarddeviation$).

During the evaluation of this module, we found that this equation corresponds to a 75% one-tailed test, i.e. 75% of predictions should be lower than this error %. The 0.674 figure is valid for more than 120 data instances, which corresponds to 30 hours of data. e.g., if mean error is 7.819% and the standard deviation of the error is 6.411%, the threshold would be: Threshold=(7.819)+ 0.674*(6.411)= 12.14.

2) Timed Build: is for the initial implementation phase of the system. It addresses the potential for improvement of the prediction model, whereas the error checking prevents degradation of the quality of predictions. The timed interval begins at 15 minutes and increases by 50% every time a re-select flag is sent due to the time interval being exceeded. The timed interval between re-selections is reset if a re-select message is sent due to an error check or a new source detected.

3) High error detected: criteria checks for deteriorations in the quality of the predictions such as a failure of a data source or other error. This uses the previous formula, with 10% of the time the prediction model is in place: (expected MAPE) + 2*(standard deviation).

4) New source detected: is activated if a listener picks up a weather observation from a new source. This adds the source to the pool of available options sooner than otherwise waiting for re-select cycle and would be very useful if a new dataset was added to the triple store in bulk with the addition of a new source.

C. Source selector

This component evaluates the sources and builds a prediction model from the best available IoT and open data sources. It then sends a *SelectionResult* object containing the prediction model, sources to be used, MAPE, RMSE, and the time it was created.

The main pillar of this component is the machine learning algorithm that it uses. We identified a set of requirements for choosing the right machine learning algorithm. These requirements are listed below in descending order, from highest to lowest priority.

- *Reasonable accuracy*: The model should generate accurate predictions.
- *Quick prediction model generation*: The model should be quickly generated.
- *Work well with little data*: The system should be able to be deployed and quickly make accurate predictions. This requires the system to not over-fit to the training set and make drastically incorrect predictions.
- *Work well with nominal and numeric inputs*: Both nominal and numeric data will be used as inputs and need to be handled by the prediction model.
- *Generalisation outside of the available training data*: This is important as the prediction model will be used in a real-time system where data encountered will often be outside the range of data in the training set.
- *Low configuration*: Low configurations are required for a portable system.
- *Low pre-processing of data*: Pre-processing of the data requires a skilled user to setup. This is often automated when using a software suite.
- *Give insights into the factors influencing the prediction*: Dependency analysis is generated for user information and understanding.

With respect to these requirements, we carried out a comparison of the common machine learning algorithms found in the literature in order to chose the most suitable one. The comparison is shown in Table II. The performance of these algorithms is discussed in Section VI-C.

The source selector needs to be scalable and efficient in its operation. The technical challenges this module faces are high memory use, processing time and latency. As the system generates multiple prediction models using training sets that potentially can span a very large time, care is required to drop references to datasets as soon as they are not needed so garbage collection and memory freeing can take place.

The processing time should be kept low by selecting sources with a greedy-type method. Evaluation of data sources is initially done over a large number of sources with small datasets and changes to more comprehensive evaluation for fewer sources. As such, reduced effort is spent on poor data sources. Latency (from queries to a remote data store) is addressed by only querying the triple store for data that is necessary and reducing duplicate queries where possible. Prediction models are built using the following attributes; power reading, time, day of week, temperature, pressure, humidity, wind speed, and wind direction.

TABLE II. COMPARISON OF MACHINE LEARNING ALGORITHMS

	Multiple linear regression	**Artificial Neural network**	**Regression tree**	**Kernel regression analysis**	**Support vector machine**
Gives insight into input importance	Yes	No (sensitivity analysis possible [8])	Yes (tree shows which variables are important)	Yes	Yes
Over-fitting prevention	Not prone to over fitting	Methods available	Pruning to stop over fitting may be required	Not prone to over fitting	Not prone to over fitting
Ease of implementation	simple	Requires manual tuning of nodes and layers	simple to understand and implement	Moderate	Moderate - standardised implementations exist
Computational cost	low	Typically high. Depends on training function	low	Depends on training function	High on large data, scales $O(n^2)$ to $O(n^3)$ (adaptations available[18])
Other benefits	Simple and quick	De-facto solution for regression on non-linear data. Extensive literature	Simple and quick	works well outside of training data range- smooth kernels	works well outside of training data
Disadvantages	Poor with non-linear relationships	Prediction outside of training data can be drastically incorrect (corrections exist for this). Unimportant inputs may worsen predictions.	Predictions not in a continuous range-binned values	needs normalizing of input data	needs normalizing of input data

V. TECHNOLOGY CHOICES AND DEPLOYMENT

A. Open Data Management

1) Data Collectors: For the sensor data collectors, we used existing data collectors [19] in the same settings. However, this work focuses more on developing data collectors for open weather data that query web addresses and receive back XML or JSON objects. The collected data is then broadcast to be further enriched. The data collectors developed for this work are:

- **NUIG Weather Station Collector** Weather is recorded as a CSV file onto an ftp server (in campus). A runnable Jar file queries this every minute and publishes the result as CSV to the JMS server.
- **OpenWeatherMap Station Collector** This station is not a physical station, but is estimation from the forecasting model made by OpenWeatherMap.org, which is then corrected with readings from other weather stations. This is accessed with an http request to a Json API every minute. The result is parsed and forwarded as a CSV text message to the JMS server.
- **YR.no Forecast Collector** A request is sent to the correct URL and a forecast is returned as Json. The result is parsed and forwarded as a CSV text message to the JMS server. As the Forecasts are created approximately every 12 hours, we decided to query this service every 6 hours.
- **Ham Weather Forecast Collector** Similarly to the yr.no generator, the Ham Weather Collector publishes new forecasts every 6 hours. The API in this case returns XML data which is parsed using a SAX feed parser.

2) Data Converters: The data converters receive CSV data and convert to RDF/XML using templates according to the chosen ontologies. The ontologies are validated using the W3C online validation service [1].

3) Data Storage: We use Virtuoso [2] a triple store. For each type of data, there is a runnable Jar with a JMS Listener for data storage. On receiving the RDF data it persists it to the appropriate Virtuoso graph.

[1]http://www.w3.org/RDF/Validator/ , accessed on: 06/06/2014

[2]http://virtuoso.openlinksw.com/

B. Source Selection and Predictive Analytics

We discuss, in the following, the development choices that we made for optimal system configuration.

1) Reduction of sources: In order for the system to be quick, scalable and memory efficient, a greedy method is used. Prediction models are built for an increasing period of time, starting with the most recent data available. Poorly performing data sources are eliminated in rounds, with a number of sources eliminated each time.

Predictions models are built for all sources initially using one day data. The dataset size is doubled for every subsequent iteration. Datasets are randomised and split into training and tests sets in the ration 90:10, and are judged on the RMSE found in the test set. Options for reducing the number of sources in each iteration could have been a fixed percentage or for fixed learning time (which would aid scalability). These would have implications for accuracy but were ultimately not tested due to a lack of available appropriate sources.

2) Data set size limits: When the data sources are reduced to two, more data is added until the RMSE is no longer decreasing. Due to the randomisation of training and test datasets, this may give less accurate results with small datasets. This will be counteracted with small datasets improving rapidly past one day.

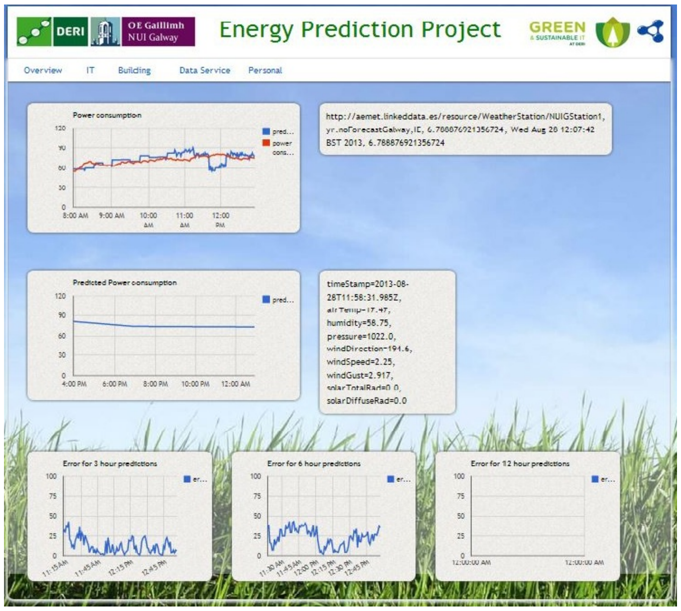
3) Forecast selection: As the prediction model is built using data from a weather station, a forecast source should provide weather predictions as close to values recorded by the selected station as possible. This is calculated with a linear correlation of data from each forecast source to the selected weather station using Pearson correlation. However, a weighted forecast selector would be better such as that implemented in [6], [20].

4) Memory use: The prediction model generator was tested on two machines, one with 8Gb memory and one with 2Gb memory. With the 8Gb system, the prediction model generator used 1Gb memory. On the 2Gb system approximately 200Mb memory was used. Informal tests did not indicate that additional memory allocation was necessary as the running time was roughly similar. The model generation is also single threaded as building multiple models at once would increase memory usage. With total running time being less than one minute for a week data for two weather stations, multithreading was not further investigated.

5) Machine Learning: We decided to use a third party library instead of developing and testing an algorithm from scratch. This makes use of the work and experience of specialist groups with thoroughly tested solutions and allows for easy comparison between algorithms. The options for this include native java libraries and libraries in other languages via wrappers. The WEKA Machine learning software set is selected because of its simple API and its active community. WEKA is maintained by the University of Waikato, New Zealand [21].

C. User Interface

The UI is implemented using the GWT client side tools. The data on the page is updated using timed remote procedure calls to the error comparison module. Fig. 3 shows the Interface of the system. The top row contains a graph of the current power consumption and what the prediction model would predict for the current conditions. This gives a rough indication of the performance of the prediction model and gives confidence to users. Also in the top row are the sources currently being used for predictions and details about the prediction model, such as the time it was created and the MAPE. The second row shows the predictions for three, six and twelve hours in the future and beside that, the current weather conditions. The bottom row shows the error rates for the three, six and twelve hour predictions.


Energy Prediction Project
Overview
IT
Building
Data Service
Personal
Power consumption
pred...
power cons...
Predicted Power consumption
Error for 3 hour predictions
Error for 6 hour predictions
Error for 12 hour predictions
er...

```
http://aemet.linkeddata.es/resource/WeatherStation/NUIGStation1,
yr.noForecastGalway,IE, 6.788876921356724, Wed Aug 28 12:07:42
BST 2013, 6.788876921356724
```

```
timeStamp=2013-08-
28T11:58:31.985Z,
airTemp=17.47,
humidity=58.75,
pressure=1022.0,
windDirection=191.6,
windSpeed=2.25,
windGust=2.917,
solarTotalRad=0.0,
solarDiffuseRad=0.0
```

Fig. 3. The User Interface of the System

VI. Evaluation and Discussion

The proposed approach has been tested on a real world use case realized in the Digital Enterprise Research Institute (DERI). The building has been retrofitted with energy sensors to monitor the consumption of power within the building. In total there are over 50 fixed energy consumption sensors covering various spaces of the building along with over 20 mobile sensors for devices, light and heaters consumption. A building-specific aspect of the dataspace has been presented in [22] with a sensor network-based situation awareness scenario.

We run three experiments for evaluating the autonomous aspect of the designed system as well as the machine learning algorithms.

A. Experiment 1 - Initiation of the System

The first experiment investigates the accuracy of the system after initial installation, i.e. with no historical weather or electrical power data. Errors in the predictions vs. actual power readings is observed over time (errors for the three, six and twelve hour-ahead predictions). Due to space limit, we discuss here only the three hour-ahead predictions plotted in Fig. 4.

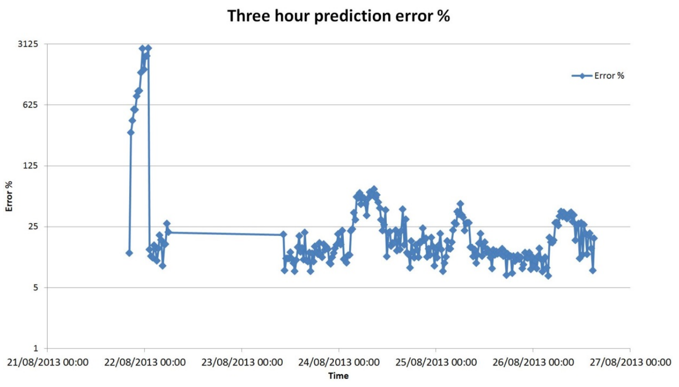


Fig. 4. Three hour prediction error vs. time

It is quite obvious that the system would provide predictions with high error rate at its initiation as there is not enough historical data for creating an accurate prediction model. This is shown on Fig. 4 by the first high error rate between 21st and 22nd of August. In addition, high error rates were observed around midnight on the 22nd of August. This corresponds the "DayOfWeek" attribute newly introduced, as only one day was in the training set at that point, the prediction for a new day was not handled well by the prediction model. Predictions for the new day hovered around 0%, and due to high percentage errors are calculated, it caused errors percentages in the 1000s. Errors quickly fell when the training set was updated with data points from the new day.

During testing there was an error with the messaging server connection for sending errors to the Build Trigger between 5am 22/08/13 and 10am 23/08/13 which caused the almost stable error rate during those days (no new data, no new prediction model). Power readings and weather observations were still collected during this time and the timed builds of the prediction model were still in effect.

The error rises on the 24th of August due to encountering the first non-working day (Saturday). On non-working days, the power consumption is lower and varies less than days when the building is at normal capacity.

After reducing over the weekend, the error % rose on Monday the 26th as the prediction model adapts to a working day after two non-working days.

B. Experiment 2 - Partial Failure of a Weather Station

Between the 23/08/2013 16:24 and 26/08/2013 18:11, the Data source selected for making prediction models was split between the NUIG weather station and the OpenWeatherMap weather API in the ratio 64:36. This shows that the data taken from the station in very close proximity to the building was slightly more preferable to the data from the weather API, though both are quite similar.

For this experiment, the system was running for 12 days to reach a steady error rate and settle on weather sources. The weather station being used for prediction model generation was the NUIG weather station and had remained that way for several days. The data collector for the NUIG weather station was altered to provide 0.0 for all temperature readings and the time for the other weather station to be selected was measured. The temperature reading is chosen to be altered as the literature shows that temperature is one of the strongest influences for Short-Term Load Forecasting [6], [20].

TABLE III. EVENT LOGGING DURING THE EXPERIMENT

Data and Time	Event
21/08/13 12:00	Fault introduced.
21/08/13 14:28	Re-selection occurred, NUIG weather station still used.
21/08/13 15:47	Switched to the other available weather station and maintained selecting it for future prediction models.

Table III reports on the time and date of the events that were observed during this experiment. We notice that the system took almost 4 hours for the prediction models to be built from an alternative weather observation source after a fault was incurred. In this time, a re-selection occurred where the source with the induced error was still used as the basis for the prediction model. This could be due to the randomisation of the data sets being used or despite there being a small amount of erroneous data, it remained better overall at that time.

C. Experiment 3 - Machine Learning Algorithms Testing

Four Machine Learning algorithms in WEKA [21] were tested for short (1 week) and long term data sets (five weeks). The datasets were for building mains incoming power and the weather observations of the NUIG weather station. For both datasets, the training set comprised of 66% of the available data and the remainder was used as the test set for evaluation. The data sets used for the testing contained the same attributes as the implemented system i.e. 15 minute averages of power reading, time, day of week, temperature, pressure, humidity, wind speed, and wind direction. The datasets were randomised and each experiment was performed three times with average values taken. Unless stated, all configurations (see Table IV) are the defaults chosen by developers of the WEKA library ver. 3.7.3.

TABLE IV. ALGORITHM CONFIGURATIONS IN WEKA FOR TESTING

Config. 1	SMOReg. The WEKA implementation of a support vector machine for regression.
Config. 2	One hidden Layer back propagation ANN with default WEKA values.
Config. 3	Two hidden layer back propagation ANN with default WEKA hidden layer one and 10 nodes in hidden layer two.
Config. 4	Linear Regression. Default WEKA implementation for multiple linear regression.

The result of the evaluation of the machine learning algorithm using short and long term data sets are respectively shown on Table V and VI. We notice from these tables that the Neural Networks, though more accurate, were far slower than the other two learning algorithms, and were not used

TABLE V. MACHINE LEARNING TESTING RESULTS FOR ONE WEEK [DATASET SIZE= 672, TRAINING DATA SIZE= 443, TEST DATA SIZE= 229].

	Mean Absolute Error (kW)	**RMSE (kW)**	**Time (s)**	**Correlation coefficient**
Config 1 (SMOReg)	5.3638	6.9759	0.851	0.6233
Config 2 (1 Layer ANN)	2.7606	3.6242	47.004	0.9158
Config 3 (2 Layer ANN)	3.0473	4.1506	50.842	0.8961
Config 4 (Linear Regression)	4.8586	5.9283	0.759	0.7396

TABLE VI. MACHINE LEARNING TESTING RESULTS FOR 5 WEEKS [DATASET SIZE= 3298, TRAINING DATA SIZE= 2176, TEST DATA SIZE= 1122].

	Mean Absolute Error (kW)	**RMSE (kW)**	**Time (s)**	**Correlation coefficient**
Config 1 (SMOReg)	4.6755	6.5965	32.4	0.8054
Config 2 (1 Layer ANN)	3.3332	4.5841	229.7	0.9162
Config 3 (2 Layer ANN)	3.7566	4.7279	247.0	0.9221
Config 4 (Linear Regression)	4.7579	6.0173	2.4	0.8396

in the system. The Linear regression and SMOReg took approximately the same amount of time, with Linear Regression being more accurate. This may be due to the homogeneity of the summer dataset not presenting non-linear trends. For the implemented system, SMOReg was chosen due to the documented ability to handle non-linear and data outside of the training set well, despite the inferior results from testing. The ANNs were initially experimented on hourly power and weather readings for 3 months from October to December, where different combinations of hidden nodes were tested. During this testing, the second hidden layer with 10 nodes was found to improve the RMSE by 1% over the single layer ANN. This improvement did not carry over to the more granular data in the real system, where adding the second decreased the accuracy of the system.

VII. RELATED WORK

Zavala et al use a numerical weather prediction model to pre-empt weather conditions and adjust heating/cooling based on the thermal resistance and thermal inertia of a building [5]. The framework proposed is based on the solution of a stochastic dynamic real-time optimization with a single weather data source.

Beccali et al use a recurrent ANN to hold the values of the hidden layer for the next time step [6] when predicting household energy consumption. Yang et al compare periodic training of ANNs with all accumulated data against ANNs trained on a sliding window of 20 days, finding the sliding window method to perform better over real data of the consumption of a building chiller [7]. Even though they found the smaller sliding window training set contributed to less processing overhead, our solution requires less data for providing acceptable results. Ismail et al compared ANNs with new methods for updating the training set [8] with large data. They used an accumulative training set, growing from the initial three months data. The sliding windows were 3, 4 and 5 months. The authors note the 5 month window yields the lowest RMSE, but they recommend the 3 month window as “the amount of data is adequate to train the model and predict accurately”.

Penya et al compare day-ahead power use in non-residential buildings for several algorithms, finding an auto regressive time series algorithm be most accurate in predicting day- ahead hourly electricity. They then used the same algorithm for four, five and six days ahead, finding the Mean Absolute Percentage Error (MAPE) to increase from 9.53% to 11.69%, 11.54% and 12.32% respectively [9].

To conclude, with respect to the evaluation of data sources, most research used just one source of weather data for short-term load forecast without assessment. Additionally, for real-time systems with continuous improvement, the majority of research used large static historical datasets for their testing while our approach does not require historical data.

VIII. Conclusion

We proposed in this paper an architecture for evaluating open data sources for real-time predictive analytics. This architecture was developed and tested on a real world case. The system is found to be achieving its goals satisfactorily. Experiment 1 shows that the system complies with the requirement of self-configuration and experiment 2 shows that the system complies with the goal of self-healing.

Experiments were conducted over summer period, with ambient temperatures within the human comfort zone (reducing the need for extra heating/cooling) hence the weather dependence may have been reduced compared to times when heating is necessary, allowing a linear combination of the time segments to approximate power use of the building.

The choice of best data source changed frequently in the running system. This could be due to the randomisation of the datasets for learning and testing in each case. Larger datasets would reduce the issue of individual points affecting the chosen source, but these larger datasets would also be more costly to process. As Experiment 2 shows, there was a slight bias towards the weather station that was directly on campus, which is expected.

As part of our future work, fully autonomic behaviour (i.e. specifying goals and the system managing itself) could be achieved with improvements in the Internet of Things. For example, searching for data sources was based on an existing work, but the Linked Data Web would enable machines discover data in an autonomic manner.

Higher accuracy of the ANNs tested was countered by large increase in computational cost over other algorithms. Methods for reducing the time taken for creating ANNs could be investigated in future, such as hybrid training schemes.

In the implemented system, bank holidays and other non-working weekdays would be treated as a regular working day and would consequently cause large errors. Web services for determining bank holidays were investigated for future work but ultimately not implemented in the current iteration.

Acknowledgement

This work has been supported in part by the Irish Research Council under the New Foundations Scheme and Enterprise Ireland, the National Development Plan and European Union under Grant Number IP/2012/0188.

References

[1] D. Evans, "The Internet of Things: How the next evolution of the internet is changing everything," *CISCO White Paper*, 2011.

[2] P. C. Zikopoulos, C. Eaton, D. deRoos, T. Deutsch, and G. Lapis, *Understanding Big Data - Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 2011.

[3] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[4] S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*, 1st ed. Wiley Publishing, 2009.

[5] V. M. Zavala, E. M. Constantinescu, T. Krause, and M. Anitescu, "On-Line Economic Optimization of Energy Systems Using Weather Forecast Information," *J. Process Control*, vol. 19, pp. 1725–1736, January 2009.

[6] M. Beccali, M. Cellura, V. Lo Brano, and A. Marvuglia, "Short-term prediction of household electricity consumption: Assessing weather sensitivity in a Mediterranean area," *Renewable and Sustainable Energy Reviews*, vol. 12, no. 8, pp. 2040–2065, 2008.

[7] J. Yang, H. Rivard, and R. Zmeureanu, "On-line building energy prediction using adaptive artificial neural networks," *Energy Build.*, vol. 37, p. 12501259, 2005.

[8] M. Ismail, R. Ibrahim, and I. Ismail, "Adaptive neural network prediction model for energy consumption," in *3rd International Conference on Computer Research and Development (ICCRD)*, vol. 4, March 2011, pp. 109–113.

[9] Y. Penya, C. Borges, D. Agote, and I. Fernandez, "Short-term load forecasting in air-conditioned non-residential Buildings," in *IEEE International Symposium on Industrial Electronics (ISIE)*, June 2011, pp. 1359–1364.

[10] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology," *IBM Corporation (October 15, 2001).*, October 2001.

[11] M. Gupta and J. Han, "Heterogeneous network-based trust analysis: a survey," *SIGKDD Explorations*, vol. 13, no. 1, pp. 54–71, 2011.

[12] F. Naumann, J. C. Freytag, and M. Spiliopoulou, "Quality driven source selection using data envelope analysis," in *IQ*, I. N. Chengalur-Smith and L. Pipino, Eds. MIT, 1998, pp. 137–152.

[13] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *International Journal of Human-Computer Studies*, vol. 43, no. 56, pp. 907 – 928, 1995.

[14] W3C Semantic Sensor Network Incubator Group, "Semantic Sensor Network Ontology," Accessed: 06-06-2014. [Online]. Available: http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[15] Ontology Engineering Group, "AEMET Weather Observation Ontology," Accessed: 06-06-2014. [Online]. Available: http://aemet.linkeddata.es/models_en.html

[16] G. A. Atemezing, Ó. Corcho, D. Garijo, J. Mora, M. Poveda-Villalón, P. Rozas, D. Vila-Suero, and B. Villazón-Terrazas, "Transforming meteorological data into linked data," *Semantic Web*, vol. 4, no. 3, pp. 285–290, 2013.

[17] Sean B. Palmer, "Meteo ontology," Accessed: 06-06-2014. [Online]. Available: http://purl.org/ns/meteo

[18] T. Joachims, "Making large-Scale SVM Learning Practical," in *Advances in Kernel Methods - Support Vector Learning*, 1999.

[19] Muntazir Mehdi and Ratnesh Sahay and Wassim Derguech and Edward Curry, "On-the-fly generation of multidimensional data cubes for web of things," in *IDEAS*. ACM, 2013, pp. 28–37.

[20] A. Cioaca, V. Zavala, and E. Constantinescu, "Adjoint Sensitivity Analysis for Numerical Weather Prediction: Applications to Power Grid Optimization," in *HiPCNA-PG*. New York, NY, USA: ACM, 2011, pp. 35–42.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[22] E. Curry, J. O'Donnell, E. Corry, S. Hasan, M. Keane, and S. O'Riain, "Linking building data in the cloud: Integrating cross-domain building data using linked data," *Advanced Engineering Informatics*, vol. 27, no. 2, pp. 206–219, 2013.