

Zodiac: Organizing Large Deployment of Sensors to Create Reusable Applications for Buildings

Bharathan Balaji[†], Chetan Verma[†], Balakrishnan Narayanaswamy[†], Yuvraj Agarwal[‡]

[†]University of California, San Diego [‡]Carnegie Mellon University
[†]{bbalaji, cverma, muralib}@ucsd.edu, [‡]yuvraj.agarwal@cs.cmu.edu

ABSTRACT

Large scale deployment of sensors is essential to practical applications in cyber physical systems. For instance, instrumenting a commercial building for ‘smart energy’ management requires deployment and operation of thousands of measurement and metering sensors and actuators that direct operation of the HVAC system. Each of these sensors need to be named consistently and constantly calibrated. Doing this process manually is not only time consuming but also error prone given the scale, heterogeneity and complexity of buildings as well as lack of uniform naming schemas. To address this challenge, we propose Zodiac— a framework for automatically classifying, naming and managing sensors based on active learning from sensor metadata. In contrast to prior work, Zodiac requires minimal user input in terms of labelling examples while being more accurate. To evaluate Zodiac, we deploy it across four real buildings on our campus and label the ground truth metadata for all the sensors in these buildings manually. Using a combination of hierarchical clustering and random forest classifiers we show that Zodiac can successfully classify sensors with an average accuracy of 98% with 28% fewer training examples when compared to a regular expression based method.

Categories and Subject Descriptors

C.3 [Special-Purpose and Application-Based Systems]: Process control systems; D.4.7 [Organization and Design]: Real-time systems and embedded systems

General Terms

Standardization, Management, Algorithms

Keywords

Smart buildings; sensor metadata; ontology; active learning

1. INTRODUCTION

Improvements in the design and manufacture of devices have led to the widespread availability of cheap sensors, actuators and data

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

BuildSys '15 Seoul, South Korea

Copyright 2015 ACM ISBN 978-1-4503-3981-0/15/11 ...\$15.00

DOI: <http://dx.doi.org/10.1145/2821650.2821674>.

collection infrastructure. This, in turn, has led to increasing interest in “Smart Environments”, which use these technologies to better understand user context and adapt to meet their requirements by controlling the physical environment around them. In pursuit of this vision, researchers have sought to create smart buildings that are responsive to occupants’ needs and comfort while conserving energy and water resources. Buildings account for 40% of the primary energy use of the US [17], emphasizing the value of saving energy resources in smart buildings.

Within commercial buildings, tasks involving indoor climate control and maintaining proper ventilation are typically performed using centralized Building Management Systems (BMS), such as Metasys from Johnson Controls [14]. A BMS interfaces with a large number of sensors and actuators deployed within buildings during construction and commissioning such as thermostats, Variable Air Volume Boxes (VAVs), Air Handler Units (AHUs), Variable Frequency Drives (VFDs) and chillers. Collectively the sensors, actuators and the BMS form an integral part of the Heating, Ventilation and the Air-Condition (HVAC) system. HVAC accounts for 48-55% of the energy consumed within buildings [35]. HVAC systems are relatively complex, typically interfacing with thousands of sensors and actuators, even in a moderately sized building (150,000 sq-ft). BMSs collect data from these sensors and provide vertically integrated tools to not only control the day to day operation of buildings, but also store and visualize data, analyze trends, and even detect faults [2, 14].

Vendors such as Johnson Controls, Siemens and Automated Logic provide proprietary tools to manage the complexity and provide different functions within buildings. These are often tied to expensive maintenance contracts and have not kept up with the state of the art in functionality, user interfaces and design. For instance, despite having fault management as a key function, facilities managers struggle to keep HVAC systems running efficiently and many faults remain unaddressed [32, 44]. Our own building managers report being notified of over 10,000 faults a day - most of which are ignored - thereby causing occupant discomfort, equipment deterioration and energy wastage [44]. This sensor management problem is compounded by lack of interoperability of methods to identify and manage sensors, and a general lack of tools to analyze large amount of sensor data generated [32, 27]. As an example, NIST estimates an annual loss of \$15.8 billion in the US due to lack of building interoperability standards [18].

Recognizing this need for systems that enable ‘smarter’ buildings, several recent efforts have attempted to address the problems of interoperability, information integration, data storage and access control. These efforts primarily propose middleware services for buildings that gather information from disparate sources of information, including a multitude of sensor protocols, and make it

available to application developers through standardized APIs [2, 5, 6, 16]. Based on our work with building infrastructure, we see that a key missing piece in all these efforts, however, is related to the assumption that the underlying sensor information is named consistently and accurately. Given the long lifetime of buildings relative to individual sensors or their networks, it is common to see sensor data information fall into disuse over time. This is exacerbated by the current practice of manually mapping sensor information for each building to a particular data model by the developers and building managers [44]. This manual mapping is expensive (requiring domain experts), time consuming, and does not generalize since it needs to be repeated for every vendor, equipment and building. The lack of standardized and automated naming has become an impediment to the creation and adoption of smart-building applications by developers that are portable across buildings and deployments.

The problem of automatically naming sensor metadata correctly and mapping the sensors and actuators to a uniform ontology is not easy. The challenges include the scale (thousands of endpoints in a moderately sized building), diverse lifetimes of buildings and BM-Ses - that are easily over 50 years in academic campuses - leading to heterogeneity in equipment types, and varying usage requirements. Researchers have identified this problem [8, 10, 39], and proposed solutions that still require significant manual effort. At the same time approaches based on using regular expressions (regex) and training examples [8, 37], do not generalize due to varying inconsistencies in sensor naming and do not leverage complementary sensor information such as its metadata and time series sensor data.

To address these challenges, we present Zodiac, a framework to analyze large numbers of sensors and actuators - including the time-series based data and the sensor metadata - and map them to a standard naming scheme with minimal human supervision. To show the efficacy of Zodiac, we applied it to four buildings on the UCSD campus comprising of over 20,000 end-points in total. To evaluate the accuracy of Zodiac we manually labeled the ground truth in terms of the sensor metadata for these sensors. We show that Zodiac classified sensor types in these buildings with an average accuracy of 98% accuracy using 28% fewer training examples, when compared to a regex based look up method, and only 15% more manual inputs than a hypothetical oracle algorithm.

The key contributions of our work are as follows.

- We show that the manual effort required to label sensors is significantly reduced through hierarchical clustering methods, without requiring customized regex that need building specific domain knowledge.
- We propose an active learning based approach that is effective in automatically identifying new sensor actuator types. Manual input is automatically requested to label these examples, and this labeling is expanded to improve coverage.
- We show that Zodiac is able to classify sensor types with high accuracy with only a small number of additional training examples than an oracle system with perfect knowledge. As compared to using regex, Zodiac uses significantly less examples and provides high sensor type classification accuracy, without requiring the significant manual effort of writing complex regexes.

We plan to release the sensor metadata to encourage researchers to develop systems that can automatically learn sensor relationships, pending permission from the university since some of the sensor data and metadata could be potentially privacy invasive.

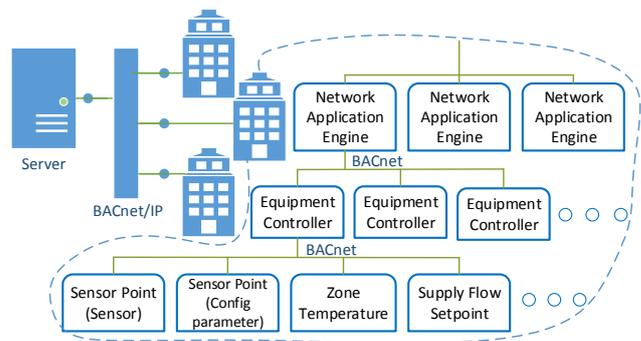


Figure 1: Figure shows the architecture of HVAC System points and the BACnet network layout leveraged for data collection

2. BACKGROUND

Modern HVAC systems consist of thousands of sensors and actuators that report information to a building control system for monitoring and maintenance. For example, a room thermostat informs the control system on how much cooling or heating is required, and helps an operator determine when the room is too hot or cold. In addition, there are configuration parameters that determine the operating point of the equipment such as cooling and heating temperature setpoints for each room. In our buildings, these serve as the higher and lower temperature bounds that the HVAC system tries to adhere to. Configuration parameters also include actuation commands such as switching ON a fan, or scheduling of equipment operation. We refer to the sensors and the associated configuration parameters in the HVAC system as *points*.

Points report data to their respective equipment *controllers*, which are embedded devices that operate the equipment control system, and react to changes in configuration parameters. Each of these controllers communicate with middle box servers, called Network Application Engines (NAEs) in Johnson Controls systems, that collect data from the controllers, and act as the interface between the HVAC system and BMS software. A subset of the NAEs in our university are connected to a dedicated network (VLAN), and they expose the points available via the BACnet protocol [9]. We have deployed our own BuildingDepot server [5] on this network to collect sensor data from 180,000 points across 55 buildings on the UCSD campus as of July 2015. (Figure 1). For this paper we focus on a subset of these buildings (four) which are of different sizes and usage modalities. In particular, for these four buildings, comprising of over 20,000 points, we had to manually label the ground truth for the sensor metadata, including their *type*. We use this labeled ground truth for both learning and testing of our automated labeling framework. The ground truth point types were based on a standardized naming template specified by UCSD contracts and our prior experience working with HVAC systems and consultation with the campus building managers. led ground truth for both learning and testing of our automated labeling framework. The ground truth point types were based on a standardized naming template specified by UCSD contracts and our prior experience working with HVAC systems and consultation with the campus building managers.

Each point in BACnet has associated metadata that describes the point and its properties. Some of these properties are specified by the BACnet standard, and others are defined by the vendor. Table 1 shows examples of six points along with a subset of their metadata. It is common for large campuses and vendors to follow a naming convention specific to the enterprise or campus [3, 4, 34]. For example, according to our university’s naming standard, “ven-

Vendor Given Name	BACnet Name	Description	Data Type	Unit	Point Type
BLDG1.N1STFLR.VAV-1NW.VAV-47.FLOW-SP	NAE-66/N2-1.VAV-47.FLOW-SP	Flow Setpoint	Analog Output	Cubic Feet per Minute	Supply Air Flow Setpoint
BLDG2.RM-2819.SUP-FLOW	NAE-14/N2 Trunk 2.VAV-35.SUP-FLOW	Supply Air Flow	Analog Input	Cubic Feet per Minute	Supply Air Flow
BLDG3 1stFl RM-111.SUPFLOW	NAE-10/N2-1.VMA101.SUPFLOW	Process Variable	Analog Input	Cubic Feet per Minute	Supply Air Flow
BLDG2.RM-1704.RM1705-T	NAE-14/N2 Trunk 2.VAV-36.RM1705-T	Room 1705 Temperature	Analog Input	Fahrenheit	Zone Temperature
BLDG3 1stFl RM-135.ZN-T	NAE-10/N2-2.VMA129.ZN-T	Zone Temperature conference rm	Analog Input	Fahrenheit	Zone Temperature
BLDG2.WBASEMENT.RM-B241.PHX-1.ZNT-SP	NAE-65/N2-2.PHX-1.ZNT-SP	Zone Temperature Setpoint	Analog Output	Fahrenheit	Common Setpoint

Table 1: Sample points from HVAC system across three buildings on the UCSD campus. Metadata of points which have the same point type have inconsistencies, and points which are different point type can have similar metadata.

“vendor given name” uses a structured format to describe a point which when split by ‘.’ gives the building name, the floor and room at which the sensor is located, the type of equipment it belongs to and the ID of the equipment, and finally, an abbreviation for the type of point. The “description” of the point gives the point type, and the “data type” gives both the type of data as well as whether it is an input or output point. As can be observed in Table 1, this naming convention is not strictly followed or enforced. The ordering of words or the punctuation may change, abbreviations and their description may change for the same point type, and as these names are entered manually per equipment, there are typographical errors and inconsistencies.

To standardize naming across buildings, we need to map the existing points into a standardized ontology [1]. We focus on accurately mapping the building points to standardized point types in this paper. Table 2 shows the number of point types for the four buildings we use for this paper. Building 1, has 3213 points that come from 154 distinct point types based on the ground truth labeling we do. Therefore, a perfect oracle algorithm that could label similar point types from a single example would still require at least 154 examples (provided by a domain expert) to label all the points in this building. Our goal is to design algorithms that can accurately label all points and require as few manual labels as possible, preferably close to unique point types in the building. Furthermore, the algorithm should be able to learn the patterns in one building, and use it for labeling points in other buildings - that is it should be able to *transfer* knowledge and labels.

As there is a naming convention based on which points are labeled on our campus and in other enterprises, regex are a natural fit for identification of sensor type [8]. As per the naming convention, the “description” of the point gives its point type, and the last part of the “vendor given name” is the abbreviation for the point type. Thus, we could maintain a mapping of description and point type abbreviations to their respective ground truth point types, and label points if their description is present in this map. As new descriptions are discovered, the domain expert is prompted to enter the point type. For Building 2, there are 922 unique descriptions mapping to 367 point types and 11910 total points. There are multiple descriptions that map to a single point type due to variations in the description, as the naming convention is not strictly enforced. The variations in descriptions can occur due to various reasons – spelling errors, additional information such as room number, or an alternate description that has the same meaning. For example, the point type “zone temperature” is also written as “zone temp”(shortening of word), “zone tempeartuer” (spell error), “room temperature” (alternate version), “zone1 temperature” or “zone temperature room 2102”(additional information). To re-

duce this variation, we remove special characters and numbers, and use uniform case. The number of unique descriptions for Building 2 reduce to 527. Some of the points in the dataset do not have any description, and we use point type abbreviation to label these points. These abbreviations can sometimes reveal the point type more accurately, as they do not necessarily vary due to changes in description. “ZN-T” is an example abbreviation of the point type “zone temperature”. However, these point types themselves vary, with use of punctuation, numbers or alternate versions of the abbreviation for the same description. The total number of unique labels with the combination of descriptions and sensor abbreviations for blank descriptions for Building 2 is 589. Thus, by some preprocessing the number of unique descriptions for 11910 points in Building 2 have been reduced from 922 to 589. It would take 589 manual inputs from experts to label Building 2, using regex to expand labeled examples. Table 1 summarizes the variation observed in descriptions and abbreviations observed across a few example point types. We design our regex to be highly accurate, and it is possible to reduce the number of manual inputs for a decrease in manual inputs. For example, regex for “zone temp” could include both descriptions “zone temperature” and “zone temp” to reduce one manual input, but may also include a false description “zone temperature setpoint”. Thus, we rely on exact matches for these descriptions. Figure 2 shows the number of points that could be labeled by regex versus number of points manually labeled for Building 1.

Among possible errors in managing sensor data are sensor naming errors. Other errors occur when descriptions of certain point types are used interchangeably across different equipment. For Building 2, 58 points were mislabeled out of 11910 total points. Note that some points are hard to label even manually because of lack of metadata, and we mark these points as “unknown”. In Building 2, 23 points were labeled as unknown.

We observe that using regex requires fairly involved domain expertise, in terms of the naming convention followed, yet can require large amounts of data. Further, regex fail to exploit additional metadata information such as unit or data type, and the actual time series of measurements, all of which can give additional clues to identify the type of point. We next describe our approach to automatically mapping each point to its sensor type using minimal manual labeling and no domain knowledge.

3. IDENTIFYING POINT TYPE

To reduce the number of manual labels required, we *group* or *cluster* the features we have for each point. We use *hierarchical* clustering to improve the grouping of points with similar metadata. Starting with a small number of labeled points, preferably

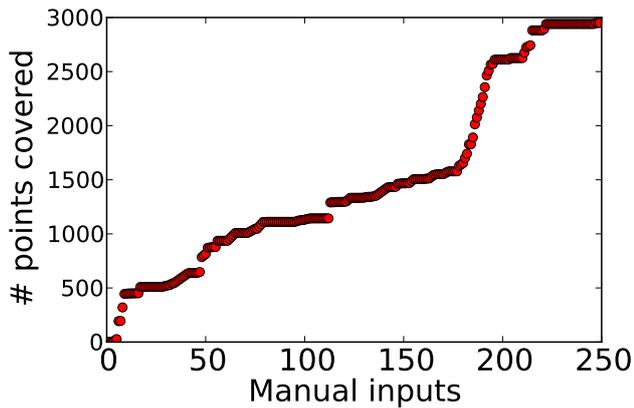


Figure 2: Number of points covered by using regex with respect to the number of sensors that were manually labeled for Building 1. Note that this graph does not account for domain expertise required to build the regex.

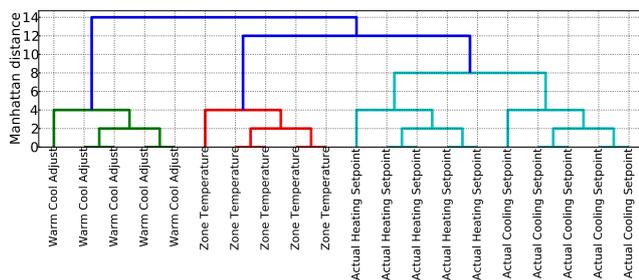


Figure 3: Example dendrogram of hierarchical clustering. Points whose metadata features are similar grouped together first and clusters which are closer to each other are consecutively grouped in the next stages.

belonging to different clusters, we train a model that automatically labels other clusters, thus achieving point label expansion. When the model determines that a (new) cluster is unrelated to any of the ones already labeled, manual labeling is requested from a domain expert for a member of this new cluster. We show that this process drastically reduces the manual effort required to assign types to points, with very few errors. Our machine learning algorithms rely on implementations from Scikit Learn [40].

3.1 Hierarchically Clustering Points

As discussed in Section 2, regex can group points which are of similar type thus reducing the manual effort in assigning point types. However writing regex requires domain knowledge to map point metadata to its type, and are dependent on the naming convention, building, and equipment providers. Clustering using point metadata offers two primary advantages over using regex for grouping. First, clustering uses the intrinsic similarities in sensor metadata rather than rely on a pre-specified pattern which may only be able to capture similarity in terms of few pre-defined descriptors. Thus, clustering can learn patterns using additional metadata such as units and data type, and can group together points which have minor variations in their metadata. As a result the grouping mechanism is more robust than an approach based on individual rules created using regex, and can generalize to a variety of naming conventions. The second advantage is that clustering based grouping of points is not dependent on domain expertise to extract useful information from the metadata. We use hierarchical clustering [23] to group points.

The features we use for hierarchical clustering are created based on the “vendor given name”, description, unit, and the type. The strings are tokenized into individual words and pre-processed to remove special characters and numbers and to convert to uniform case. A *bag of words* [45] representation is used for the feature set. Hierarchical agglomerative clustering computes the distance between a pair of points using their feature vectors, and merges those points which have the least distance between them. These clusters are then recursively merged again based on the linkage metric used. We use *complete* linkage, which combines clusters after examining each point within the cluster, and use *manhattan distance* as the distance metric. The results obtained are similar for other distance metrics such as euclidean distance and jaccard index. Manhattan distance is used for our results since it is easier to interpret in terms of the difference between point features.

Figure 3 shows an example dendrogram obtained by hierarchical clustering of 20 points for illustration. Since the metadata used for describing points of the same type are similar, the distance between their feature vectors is small, and they naturally get clustered together in the first few stages. As the clusters get bigger, points of different types also get merged eventually forming one big cluster. An appropriate threshold distance on the Manhattan distance needs to be identified that would prevent the merging of clusters with different point types.

Figure 4 shows the dendrogram obtained with hierarchical clustering of 1000 points from one of the buildings in our testbed. The horizontal lines represent choices for threshold for obtaining point clusters. As we increase the threshold, the number of points in a cluster increase, and points of different types may be clustered together. As we decrease the threshold the total number of clusters increases. As the number of clusters reflect the number of manual checks that may be required for labeling points, we would like to obtain as few clusters as possible. These trade offs can be quantified using network motif methods [20], but for this work we pick a threshold from the dendrogram based on an estimate of the number of manually labeled points. We conservatively pick a low threshold to minimize the number of errors in clustering, where errors correspond to points of different types clustering together. As the feature set we use is the same across buildings, this threshold remains the same for hierarchical clustering of points in other buildings as well.

We *conservatively* define a cluster to be erroneous if it contains points of more than one ground truth point type. So, a cluster and all the points within it are marked incorrect even if only one out of hundreds of points is included incorrectly. We applied hierarchical clustering on 11,900 points in Building 2, and obtained 1105 clusters. Only 18 of these clusters were erroneous, giving an accuracy of 99.3% with only 85 points mislabeled. An error in clustering occurs when the metadata used to describe points of different type are very similar. As an observed example, two point types “hot water pump status” and “chilled water pump status” were misclassified because their descriptions were blank, and only two letters in their entire metadata were different. It was observed that by examining the errors in clustering, it may be possible to identify errors in metadata as well. For example, point types “zone temperature” and “zone temperature setpoint” are input and output points respectively, but both of them were labeled as inputs, causing an error in clustering. As discussed earlier, the clustering was able to combine points which would have been difficult to identify as similar using regex rules. For example, points with descriptions “zone temperature”, “cold box temperature” and “freezer temperature” were clustered together as their other metadata are similar. Column (e) of Table 2 summarizes the clustering results of four buildings.

From Table 2 we see that Building 2 has 367 unique point types

Building Name	Total # Points	# Point Types	# Unique Descriptions	# Clusters	Accuracy %	Learning with Hierarchical # Manual	Learning with Hierarchical Accuracy	# Merged Clusters	Learning with Merged # Manual	Learning with Merged Accuracy
(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
Bldg 1	3213	154	251	300	98.7%	245	99.3%	191	181	98.3%
Bldg 2	11910	367	589	1105	99.3%	548	94.5%	499	453	96.0%
Bldg 3	1913	156	228	215	97.1%	204	99.8%	174	169	98.8%
Bldg 4	4380	192	316	329	98.8%	299	100%	206	198	99.1%

Table 2: Table lists the characteristics of four testbed buildings on our campus with a total of over 20,000 points. The # Point Types (c) is obtained from the ground truth labels and represents number of examples required to learn point types by an oracle algorithm. The unique descriptions (d) are obtained by manual pattern inference and represent the number of examples required to learn point types by our regex algorithm. Results of hierarchical clustering (e), and random forest based active learning (g,h,j,k) is compared with the number of manual examples required for both regex and oracle algorithm.

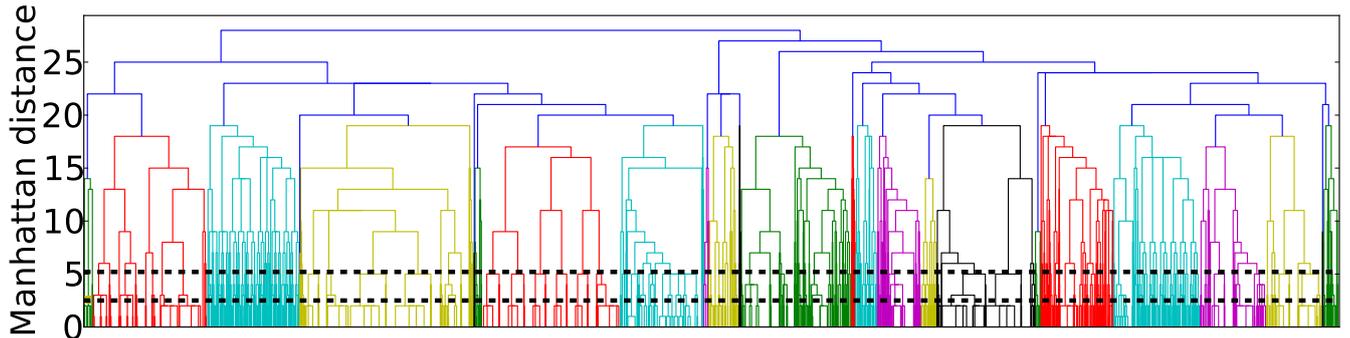


Figure 4: Dendrogram for 1000 points in a building. The two dashed lines are the thresholds that can be chosen to increase or decrease number of clusters obtained. As the threshold increases the number of clusters decreases and the accuracy of clustering points of same type decreases.

(column c), 922 unique descriptions, 589 unique descriptions (column d) after their case is normalized, numbers are removed and blank descriptions are mapped to abbreviations. Hierarchical clustering gives 1105 clusters (column e), which is more than domain based heuristics. However, the clusters obtained from hierarchical clustering capture the inherent variation in the naming structure, which is different from those obtained using domain knowledge. This is because hierarchical clustering looks for similarities across all the metadata of points while regex is based on domain knowledge that specific metadata such as point description is more indicative of the point type. As we show in Section 3.2, hierarchical clusters can be useful in learning data driven models. The low intra-cluster error generated by the hierarchical clustering can then be leveraged to efficiently label a cluster by manually assigning type of one point in each cluster and propagating the same to other points in the same cluster.

Furthermore, domain knowledge can be used to improve the clustering further. We combine two clusters (obtained using hierarchical clustering) when the description of each point across these clusters were identical. As a result of this, for Building 2, the number of unique clusters dropped to 499 (column i) compared to 589 unique descriptions. Thus, under the availability of the knowledge of which parts of the point metadata are most important, the number of lookups required can be reduced when compared to using regex. Figure 5 shows the comparison of the clusters obtained for Building 1. We next show how manual input can be further reduced by learning predictive models based on point metadata.

3.2 Learning Point Types

The regex based labeling of point types uses a look up table for

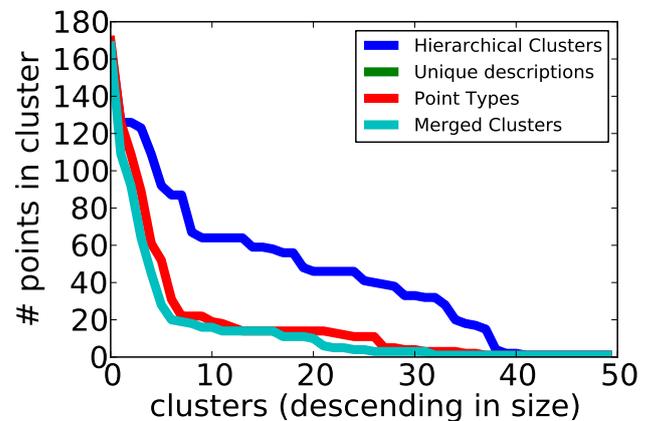


Figure 5: Histogram of points for Building 1 by unique descriptions, hierarchical clustering, merged clustering and ground truth point types. The number of clusters have been cutoff at 50 out of a maximum of 300

different kinds of metadata, and maps it to its type. Hence, it relies on exact matches on point types, and each variation of metadata needs to be manually verified before it is added to the table. Although some of this variation in metadata is captured using hierarchical clustering, it does not comprehensively capture all the variations that occur. For example, suppose the point type “supply air flow” has two points with descriptions “supply air flow” and “supply flow feedback”. A look up table based match cannot automatically learn that “supply air flow feedback” is also a description

of the same point type. Thus, there will be three separate lookups for “supply air flow”, “supply flow feedback” and “supply air flow feedback” using the regex method. If the metadata features used by hierarchical clustering is different for these three points, they would also be put into three separate clusters.

A data driven model can learn the relationship between metadata and ground truth point types, and it can give a prediction for metadata whose examples have not been observed before. Hence, a data driven model is capable of learning that “supply air flow”, “supply flow feedback” and “supply air flow feedback” belong to the same point type. Further, sensor timeseries data can be used to learn models (or rules, or regex) even when a pre-defined format is unknown for using regex. They can also incorporate additional features for learning the characteristics of a point. We present our timeseries data based features used for learning a model in Section 3.3

To validate our hypothesis that it is possible to learn an effective model that can use sensor metadata to predict the point type, we micro benchmark the performance of a Random Forest classifier [29] for Building 1. We use three fold cross-validation with the training set having at least one example of each point type. We observe that the Random Forest classifier can successfully identify point types with an average accuracy of 97.1%.

A key challenge for training a model that can predict point types based on their metadata is the availability of labeled training points. In order to train a mapping model with minimal manual input, we use *active learning*. To begin, we seed the learning with ten points which have been labeled by a domain expert. When we inspect the next point, we need to identify if this point is of the same types as the ones we have already have a label for, and if not, ask the domain expert if it is a new type.

There are two conflicting forces in learning such a model. There are point types which have metadata, i.e., features, which are very close to an existing point type, but it is of a different type. For example, “occupied temperature setpoint” and “unoccupied temperature setpoint”. For a poorly trained model, these will be misclassified. And then there are points whose features are very different, but actually are of the same type. For example, “airflow rate” and “supply flow feedback”. The model may mark one of these as a new type, and hence, it increases the number of manual inputs required.

To check whether an unseen point is of a new type, we can assign a probability of the point belonging to one of the existing classes (i.e., point types). We can build a generative model for each of the point types seen so far, and use this model to assign this probability. We experimented with Gaussian Mixture model [36], Multinomial Naive Bayes model [25] and one class SVM [31], and they did not work well with our dataset. A discriminative model on the other hand would give the probability of a point mapping to one of the existing classes. If the model assigns low probability to all the existing classes, then there is a good chance that this is of new point type. In line with this intuition, the Random Forest classifier [29] consistently gives low probability to a point of new type in our dataset. Hence, we use the Random Forest classifier to determine if we need to ask the domain expert for the correct label.

To label the points of a building, we first cluster the points using hierarchical clustering, and assume the points within a cluster are of the same type (Section 3.1). We ask for labels for 10 randomly chosen points from distinct clusters from the domain expert, propagate these labels to all the points in their respective clusters and build a Random Forest classifier based on these points. Next, we obtain the prediction probability for a new point. If its probability is high (>0.9), we assume the prediction to be correct, and add the points in the corresponding cluster to the training set. If the proba-

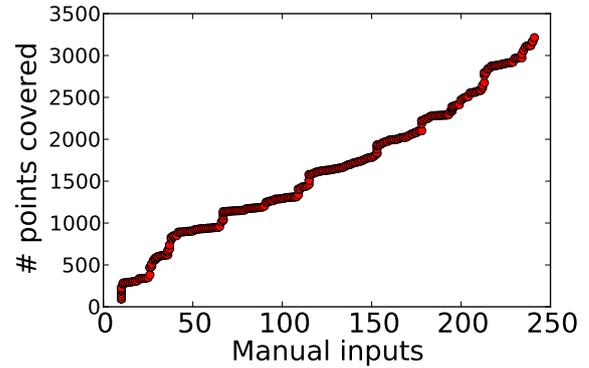


Figure 6: Learning curve of random forest based active learning algorithm. It required manual labeling of 245 points for labeling 3213 points in Building 1 with accuracy of 99.3%.

bility is low (<0.2), we ask the domain expert for the point type. We iteratively retrain the model using the labels learned, and add more points to the cluster. When there are no more points which satisfy the upper/lower probability thresholds, we decrease/increase the thresholds respectively, to learn more points.

Figure 6 shows the results for the random forest based active learning for Building 1. The number of manual inputs required for all points of Building 1 is 245, and the accuracy of labeling with the obtained random forest classifier is 99.3%. Thus, our random forest based active learner is able to learn the mapping of points in Building 1 with 6 fewer examples than regex methods (251 examples, Figure 2) without any prior knowledge about the structure of the naming convention. When we used the merged clusters obtained by combining unique descriptions and hierarchical clusters (Section 3.1), the number of manual examples required dropped to 181, with an accuracy of 98.3%. Columns (g, h, j, k) of Table 2 summarizes the results for active learning methods on four buildings. A limitation of our algorithm is that learning rate with manual inputs is linear as seen in Figure 6. However, a better algorithm could be devised that takes advantage of frequently occurring point types (Figure 10) to increase the learning rate.

3.2.1 Learning Across Multiple Buildings

With regex and look up tables, it is easy to use mapping from one building to learn the mapping of another. Some point types such as “zone temperature” and “supply air flow” are common across buildings, and once their mapping is learned, points of the same type in other buildings can be labeled. To test how much information can be learned across buildings using regex, we created a look up table using ground truth point types of Building 2 and used it to label points of Building 1. Figure 7 shows the learning curve obtained for Building 1. All the points in the building were learned using 176 manual inputs, a reduction of 75 labels compared to regex based learning without any prior knowledge. However, as the description of the two buildings do not follow the exact same terminology, some errors are introduced, and the accuracy drops to 99.6%.

The active learning method used by Zodiac also learns the mapping between point metadata and its type, it should be able to label points of the same type even across different buildings as their metadata will be similar. To evaluate the *transfer learning* capability of Zodiac, we first built a random forest classifier using ground truth point types of Building 2, and used the iterative learning method to label points of Building 1. We again used 10 randomly chosen points as seed examples, and the feature vector for the classifier consists of bag of words of point metadata from both buildings. Figure 8 shows the learning curve obtained for mapping

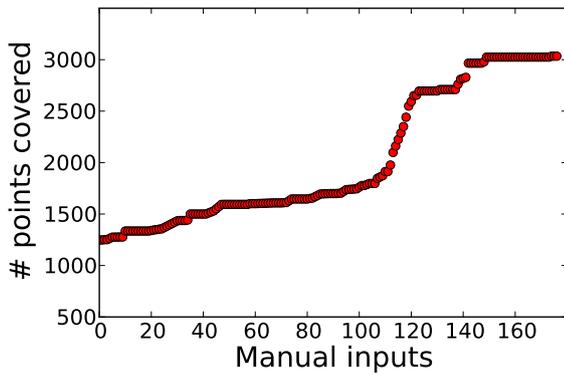


Figure 7: Learning curve of regex based naming across multiple buildings. It requires manual labels for 176 points in Building 1 (total 3213 points) with accuracy of 99.6%.

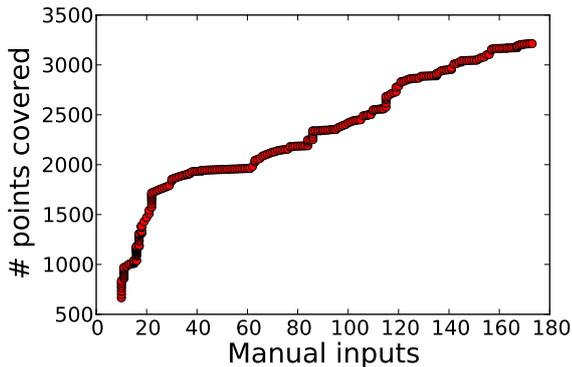


Figure 8: Learning curve of random forest based active learning across multiple buildings. It required manual labels for 173 points in Building 1 (total 3213 points) with accuracy of 99.5%.

of Building 1 points. 173 manual inputs were used for learning, an improvement of 51 manual labels compared to learning without any prior experience. Thus, the active learning method is able to successfully learn from prior experience without domain knowledge, and is about as successful as the regex method. The accuracy of classification was 99.5%, and hence, the learning model is able to label points using information from another building. This is an initial result based on one example, and we are in the process of evaluating transfer learning across other buildings.

3.3 Using Time Series Data

To improve the accuracy of point labeling and to further reduce the manual input, we next try and leverage the time series data from points. This highlights an advantage of our learning based model, as it can incorporate any additional information that is available.

In general, time series can be divided into episodes, where each episode is a sample. For example, in many sensor applications, such as building HVAC, there is a natural diurnal variation leading to day length episodes. In order to leverage the time series data, the problem we are faced with is one of time series classification [19]. This is a problem that has been well studied in the data mining literature, see for example [24, 30] for representative techniques and applications. However, these methods often exclusively focus on coarse patterns or motifs that can be used to distinguish time series generated from very different processes. In our case, due to the fact that most points are associated with a common HVAC process - with common diurnal variation and dependence on external temperature - many time series will have similar patterns. Time se-

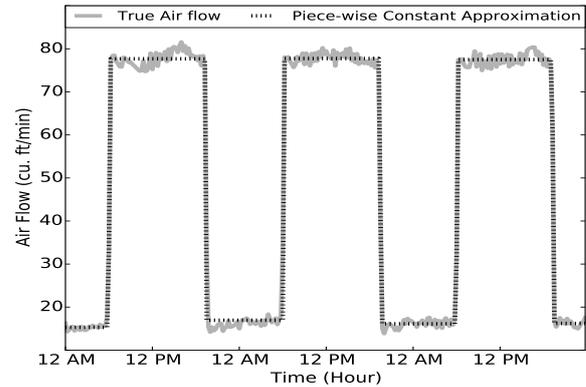


Figure 9: Piece-wise constant approximation of time series

ries classification based on fine grained time series features arises in applications like speaker recognition [7] and signature based appliance classification [22]. In our experiments we combine features that capture many levels of the time series structure.

We use four classes of features, namely, scale based, pattern based, texture based and shape based features. *Scale based* features capture the range of values that the point readings can take. We use mean, max, min, upper and lower quartiles and range. For example, the mean and range of sensor measurements can tell us if a sensor measures supply air temperature or supply water temperature. *Pattern based* features capture the structure of repetitive sub-patterns in the time series. We use three Haar wavelet and three Fourier coefficients from the power spectral density of the signal as features. *Shape based* features capture the coarse structure of the time series, but are insensitive to fine structure. We use a piece-wise constant model of the time series, as shown in Figure 9, and use the location and magnitude of top two components as features. The error variance between the piece-wise constant model and the true signal is used as a texture feature. *Texture based* features capture the roughness, smoothness and other fine scale features of the time series. Texture based features have a history in image processing applications, but have found limited application in time series classification. However, we find that the texture based features we use are particularly useful in distinguishing between points like supply flow - which is rough - and their corresponding set-points which are smooth. In addition to the error variance mentioned above, we use the variance of the difference and second difference between consecutive samples, max variation, number of up and down changes along with an edge entropy measure. The edge entropy measure is intended to capture the regularity of the time series across multiple episodes (a day in our case). For each episode, we capture the times at which large changes in value occur, and accumulate these as counts across episodes. We normalize these counts to sum to one, and compute the entropy of the resulting probability distribution. If this entropy is high, we can infer that the point either has limited structure within each episode or between episodes - a useful feature for point data classification.

As described, we select six features of each type, for a net total of 24 data dependent time series features. While the selection and design of more application specific features may be useful, we find that these features perform well in practice. To test this, we evaluate the *time series data only*, *metadata only* and *time series data+metadata* features on a separate building with 5857 points divided into 198 unique types. The distribution of point frequencies is shown in Figure 10.

The accuracy of the three methods, as a function of the number of labeled points of each type available is shown in Figure 11. We observe that the time series or data based features are not very ef-

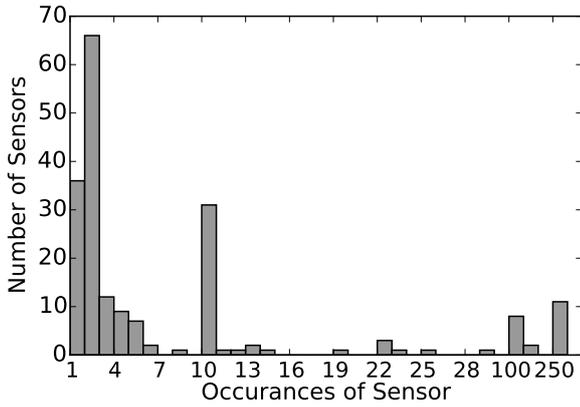


Figure 10: Relative occurrence counts of different point types in a building

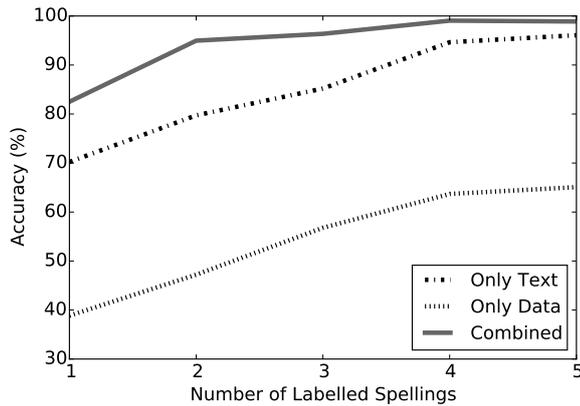


Figure 11: Comparison of text (i.e., point metadata) and (time series) data based learning methods. The x-axis represents the number of labels required for each point type. We observe significant improvement when both metadata and time series data features are used together.

fective on their own. This is because of two reasons. There are some point combinations ‘supply air flow setpoint’ and ‘cooling minimum flow’ which are essentially identical as time series in the way our buildings are configured. Secondly, there are many points - such as some set points and heating commands that never change (are always 0) hence are again impossible to differentiate using data alone. Finally, there are some points like ‘Heating Command’ and ‘Cooling Command’ that are very similar at coarse and fine time scales (taking values between 0 and 1, sharp changes at apparently arbitrary points) that are essentially impossible to distinguish using the features we use. We note that they can be distinguished using point inter-relationships (heating command will be high when zone temperature is too low), and does suggest a direction of future work. However, in Figure 11 we do see that using point data is able to provide significant boost to accuracy over using point metadata alone, particularly when only a few labeled examples are available - exactly the regime that is of interest to us. This demonstrates potential of using additional information such as point time series data to improve the classification models.

We add the time series data features to the metadata features to test if they help our active learning algorithm. Figure 12 shows the learning curve of the active learning algorithm for Building 1 with 3213 points and 300 hierarchical clusters. The additional data features lead to a slight drop in accuracy (98% vs 99.3%) and an increase in manual examples required (261 vs 245). We observed

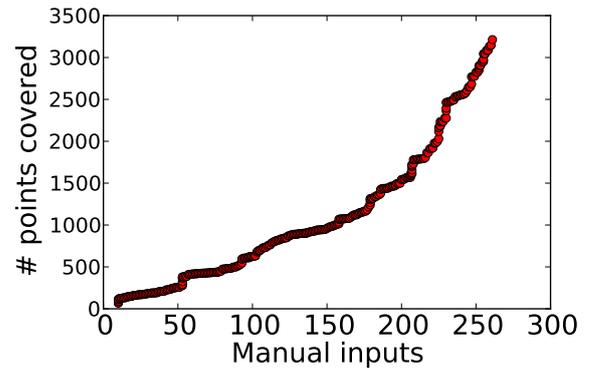


Figure 12: Learning curve of random forest based active learning algorithm with additional features extracted from time-series data for Building 1 with use of hierarchical clusters. It required manual labeling of 261 points for labeling 3213 points in Building 1 with accuracy of 98%.

similar results when we used merged clusters for Building 1. As the random forest based active learning algorithm is based on confidence of label classification, the result indicates that the added data features led to ambiguity and decreased the confidence threshold. Hence, additional manual inputs are solicited until lower thresholds can be included. It is possible that a better active learning algorithm can incorporate data features with improved accuracy. Moreover in the absence of point metadata, the point time series data can be leveraged to reduce manual labeling.

4. RELATED WORK

The organization of sensors using standardized ontologies has been recognized in literature as a key component for building useful, reusable applications [13, 38, 43]. OntoSensor [38] is a system that labels sensors using an ontology and describes them using a UML like language for representation and querying. Within the buildings domain, mapping of sensors, or points, to standardized ontologies is considered to be critical for information re-usability and development of apps that improve energy efficiency [10, 21, 26, 37]. Standards are being developed for naming of points in the HVAC system [1, 26], and system architectures have been proposed that build upon a standardized information ontology to build applications that understand contextual information [21]. However, none of these works focus on mapping of existing points in a building to a standard ontology.

Schumann et al. [39] identify the mapping of existing points to a standard format to be a challenge, and propose artificial intelligence based methods such as hierarchical clustering for learning such a mapping automatically. However, they do not implement or evaluate their proposed method on real sensors. Reinisch et al. [37] propose a platform that facilitates the mapping of points to a standard ontology. They do not, however, learn the mapping, and still rely on manual inputs. Bhattacharya et al. [8] address the problem of organizing points to a standard template, and their work is closest to our work. They use a synthesis technique that constructs a metadata structure using transformation rules, and evaluate their technique on point metadata from several building in their university. We propose a different approach to the same problem, with use of learning based methods. The advantage of using a mixture of hierarchical clustering and active learning based methods is that we can use known information about the points to learn their model. For example, we used information such as unit, data type, and characteristics of data variation as features in our model. In contrast, the approach taken by Bhattacharya et al. [8] requires a

human to recognize and formulate the patterns to identify sensor point types.

In machine learning terminology, the learning paradigm where both labeled and unlabeled points are available is called semi-supervised learning [11]. Active learning is a form of semi-supervised learning where the learning algorithm presents unlabeled points to an expert who returns labels for them [12]. It is expected that efficient querying algorithms will require fewer labeled samples. We consider pool based active learning algorithms, which exploit situations where a small set of labeled data and a large pool of unlabeled data are available [28]. Our active learning algorithms are uncertainty based - that is we query points we are least confident about. However, we modify these algorithms to be partly density based [33], i.e., we select which points to query based on cluster sizes. Based on the intuition that sensors in buildings and related large scale applications are spatio-temporally organized, we incorporate ideas from hierarchical active learning [15]. Finally, we note that the use of random forest ensembles with uncertainty based sampling is related to the idea of Query-by-Committee [42]. While many variants of active and interactive learning have been proposed in the literature [41], we find that the algorithms we have chosen work well in practice for our application. Investigating other alternatives remains an interesting direction for future work.

5. LIMITATIONS AND FUTURE WORK

We have shown that it is possible to learn the naming patterns in HVAC systems, and classify points according to their types with minimal supervision. We have tried our methodology across four buildings with promising results. Our dataset, however, is limited to the UC San Diego's university campus, and most of the equipment is installed by one vendor. The point naming standards used across many different institutions are similar to ours [3, 4, 34], but it remains to be seen how our algorithm will generalize to a different set of equipment, vendors and facilities management.

Standardized point names are an important step towards portable smart building applications, however there is additional domain specific context that is not captured by uniform naming. For example, points need to be categorized according to the equipment they belong to, and the type of equipment needs to be identified for applications like fault detection and energy analysis. With our text metadata from BACnet, we used equipment specific features to identify equipment ID and equipment type. For buildings with well labeled points, hierarchical clustering successfully grouped points by equipment and clustered the equipment by their type. However, many buildings had point metadata that lacked equipment information or had poor equipment naming, and hierarchical clustering failed for these buildings because the metadata features were not adequate to cluster points by their equipment type. In future work, we will pursue methods that would map the points to their respective equipment, identify the equipment and learn the relationship between the points.

An example of such a problem is the mapping between VAV boxes and AHUs in a building HVAC system. Many buildings in our campus do not have this mapping information in the metadata, and facilities managers resort to manually maintained documents, or scrutinizing building architectural diagrams. In preliminary experiments, we attempted to find this mapping using data driven methods that identify the correlation between AHU behavior and the corresponding VAVs served by that AHU. However, the variation in temperature or airflow data was inadequate to capture this correlation. One promising approach is to use actuation of HVAC system according to a controlled sequence to learn such relationship between equipment across the building empirically.

Understanding such domain specific context and standardized representation of this context is key to developing portable applications that provide useful insights based on sensor information and provide value added services. To encourage research for development of methods that automatically learn relationship between points and map them to a standardized representation, we release the dataset consisting of metadata of 180,000 points across 55 buildings in the URL: <http://www.synergylabs.org/datasets/zodiac.html>.

6. ACKNOWLEDGMENTS

We thank Anna Levitt, Robert Austin, Bob Caldwell and John Dilliot from UCSD Facilities Management who worked with us on campus wide data collection. Thanks to Ashish Tawari for providing us guidance on hierarchical clustering and active learning methods. This work was funded by UCSD Facilities Smart Building Deployment grant and NSF grant #1526237.

7. CONCLUSION

Heterogeneity in sensor naming and metadata are an impediment to development of reusable applications in large scale sensor deployments. We illustrate the scale and challenges in mapping sensors in HVAC systems in our university buildings to a standardized naming schema. Regular expression based methods can map sensors to their respective types but tend to be too sensitive to minor variations in the sensor metadata and require substantial domain expertise. Our proposed framework, Zodiac, uses hierarchical clustering for grouping the sensors based on the inherent patterns in the sensor metadata. We applied hierarchical clustering on four buildings in our campus, and the clusters were grouped together based on sensor type with an average accuracy of 98% as compared to the manually labelled ground truth. Zodiac uses the clusters to train a random forest classifier using active learning. Our active learning algorithm labeled sensors across four buildings to their respective types with an average accuracy of 98% requiring 27% fewer ground truth labels than regular expression based methods.

8. REFERENCES

- [1] Project Haystack. <http://project-haystack.org/>.
- [2] Tridium Niagara AX. <http://www.niagaraax.com/>.
- [3] Building Automation System Design and Construction Standards. *Energy Operations Group, Utilities and Energy Management, University of Rochester*, 2013.
- [4] Unified Facilities Guide Specifications. *Department of Defense Unified Facilities Criteria*, 2015.
- [5] Y. Agarwal, R. Gupta, D. Komaki, and T. Weng. Buildingdepot: an extensible and distributed architecture for building data storage, access and sharing. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 64–71. ACM, 2012.
- [6] P. Arjunan, N. Batra, H. Choi, A. Singh, P. Singh, and M. B. Srivastava. Sensoract: a privacy and security aware federated middleware for building management. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 80–87. ACM, 2012.
- [7] M. Basseville. Distance measures for signal processing and pattern recognition. *Signal processing*, 18(4):349–369, 1989.
- [8] A. Bhattacharya, D. E. Culler, J. Ortiz, D. Hong, and K. Whitehouse. Enabling portable building applications through automated metadata transformation. Technical report, Technical Report UCB/EECS-2014-159, EECS Department, University of California, Berkeley, 2014.

- [9] S. T. Bushby. BACnet™: A standard communication infrastructure for intelligent buildings. *Automation in Construction*, 6(5):529–540, 1997.
- [10] J. F. Butler and R. Veelenturf. Point naming standards. *ASHRAE Journal*, (November issue):B16–B20, 2010.
- [11] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.
- [12] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.
- [13] M. Compton, C. Henson, L. Lefort, H. Neuhaus, and A. P. Sheth. A survey of the semantic specification of sensors. 2009.
- [14] J. Controls. Metasys facility management system, 1999.
- [15] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM, 2008.
- [16] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. E. Culler. Boss: Building operating system services. In *NSDI*, volume 13, pages 443–458, 2013.
- [17] E. Efficiency. Buildings energy data book. *US Department of Energy*. <http://buildingsdatabook.eere.energy.gov/>. John Dieckmann is a director and Alissa Cooperman is a technologist in the Mechanical Systems Group of TIAx, Cambridge, Mass. James Brodrick, Ph. D., is a project manager with the Building Technologies Program, US Department of Energy, Washington, DC, 2009.
- [18] N. GCR. Cost analysis of inadequate interoperability in the us capital facilities industry. *National Institute of Standards and Technology (NIST)*, 2004.
- [19] P. Geurts. Pattern extraction for time series classification. In *Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.
- [20] J. A. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Research in Computational Molecular Biology*, pages 92–106. Springer, 2007.
- [21] J. Han, Y.-K. Jeong, and I. Lee. Efficient building energy management system based on ontology, inference rules, and simulation. In *Proceedings of the 2011 International Conference on Intelligent Building and Management, Singapore*, volume 5, 2011.
- [22] S.-J. Huang, C.-T. Hsieh, L.-C. Kuo, C.-W. Lin, C.-W. Chang, and S.-A. Fang. Classification of home appliance electricity consumption using power signature and harmonic features. In *Power Electronics and Drive Systems (PEDS), 2011 IEEE Ninth International Conference on*, pages 596–599. IEEE, 2011.
- [23] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.
- [24] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, volume 98, pages 239–243, 1998.
- [25] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes. Multinomial naive bayes for text categorization revisited. In *AI 2004: Advances in Artificial Intelligence*, pages 488–499. Springer, 2005.
- [26] K. J. Lee, O. Akin, B. Akin, J. Garrett, and S. Bushby. Ontology development for low-energy building embedded commissioning. 2009.
- [27] D. Lehrer and J. Vasudev. Visualizing information to improve building performance: A study of expert users. *Center for the Built Environment*, 2010.
- [28] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.
- [29] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.
- [30] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.
- [31] L. M. Manevitz and M. Yousef. One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154, 2002.
- [32] E. Mills. Building commissioning: A golden opportunity for reducing energy costs and greenhouse-gas emissions. *Lawrence Berkeley National Laboratory*, 2010.
- [33] K. Nigam and A. McCallum. Employing em in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, 1998.
- [34] L. Olsen. Equipment Naming Convention. *University of Alberta, Facilities and Operations Design Guidelines*, 2015.
- [35] L. Pérez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.
- [36] C. E. Rasmussen. The infinite gaussian mixture model. In *NIPS*, volume 12, pages 554–560, 1999.
- [37] C. Reinisch, W. Granzer, F. Praus, and W. Kastner. Integration of heterogeneous building automation systems using ontologies. In *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, pages 2736–2741. IEEE, 2008.
- [38] D. J. Russomanno, C. R. Kothari, and O. A. Thomas. Building a sensor ontology: A practical approach leveraging iso and ogc models. In *IC-AI*, pages 637–643, 2005.
- [39] A. Schumann, J. Hayes, P. Pompey, and O. Verscheure. Adaptable fault identification for smart buildings. In *Artificial Intelligence and Smarter Living*, 2011.
- [40] scikit-learn Machine Learning in Python, 2015. <http://scikit-learn.org/>.
- [41] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.
- [42] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.
- [43] A. Sheth, C. Henson, and S. S. Sahoo. Semantic sensor web. *Internet Computing, IEEE*, 12(4):78–83, 2008.
- [44] H. Teraoka, B. Balaji, R. Zhang, A. Nwokafor, B. Narayanaswamy, and Y. Agarwal. BuildingSherlock: Fault Management Framework for HVAC Systems in Commercial Buildings. *Technical Report, CSE, UCSD*, 2014.
- [45] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.