Demo Abstract: Writing Scalable Building Efficiency Applications using Normalized Metadata

Arka Bhattacharya [*], [†]
David Culler
Dept. of Electrical Engineering
and Computer Sciences
University of California, Berkeley
arka,culler@eecs.berkeley.edu

Dezhi Hong,
Kamin Whitehouse
Dept. of Computer Science
University of Virginia
hong,whitehouse@virginia.edu

Jorge Ortiz
IBM TJ Watson Research Center
jjortiz@us.ibm.com

Abstract

Extracting meaningful information from a building's sensor data, or writing control applications using the data, depends on the metadata available to interpret it, whether provided by novel networks or legacy instrumentation. Commercial buildings comprise large sensor networks, but have limited, obscure 'tags' that are often meaningful only to the facility managers. Moreover, this primitive metadata is imprecise and varies across vendors and deployments.

This state-of-the-art is a fundamental barrier to scaling analytics or intelligent control across the building stock, as even the basic steps involve labor intensive manual efforts by highly trained consultants. Writing building applications on its sensor network remains largely intractable as it involves extensive help from an expert in each building's design and operation to identify the sensors of interest and create the associated metadata. This process is repeated for each application development in a particular building, and across different buildings. This results in customized building-specific applications which are not portable or scalable across buildings.

We have developed a synthesis technique([2]) that learns how to transform (*normalize*) a building's primitive sensor metadata to a common namespace(similar to [1]) by using a small number of examples from an expert, such as the building manager [1](Figure 1). Once the transformation rules are learned for one building, it can be applied across buildings with a similar metadata structure. This common and understandable namespace automatically yields *semantic relationships between sensors*, which enable analytics applications that do not require apriori building-specific knowledge.

In this demonstration, we present three efficiency and analytics applications — (a) Identifying rogue thermal zones, i.e zones that require constant heating or cooling (b) Identifying stuck air flow dampers, and (c) Identifying the presence of night-time setbacks — all built against our common namespace. We were able to apply these applications unmodified to more than 10 commercial buildings on the University of California, Berkeley campus. These buildings comprised sensor networks which were commissioned by two different vendors at different points in time over the last two decades. The applications helped identify candidate efficiency and comfort improvements in each of these buildings without any manual inspection.

1 Scalable Building Efficiency Analytics

The three example applciations we describe below were run on more than 10 buildings. An expert helped transform the primitive sensor metadata of two of these buildings to the common namespace through very few examples[2], and the transformation programs learned by our technique were then successfully applied to the remaining buildings in our testbed. The transformed metadata yielded *semantic relationships* between various sensors (Figure 1) in a particular building. We illustrate the results of our applications from one of the buildings in our testbed.

Rogue Zones: A zone (or room) is *rogue* if its air temperature is constantly above or below their required setpoints, i.e it requires constant cooling or constant heating. Such zones constrain supply air settings and constrain efficiency improvements. For example, a zone which requires constant heating may drive the air handler to supply air that is too cold, resulting in other zones supplied by that air handler always being too cold. To find the rogue zones in a building, one needs to simply search for sensors having the metadata tag **room air temp sensor** (in the common namespace), and for each such sensor, find a sensor with the tag **room air**

---

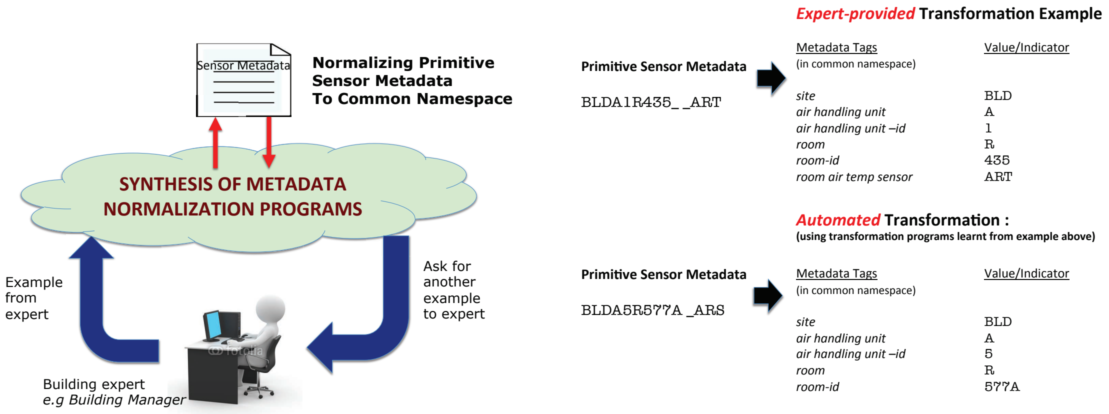
[*] Primary Author.

[†] This work was supported in part by NSF grants CPS-0931843 (ActionWebs), CPS-1239552 (SDB) and the DOE OpenBAS grant.

[1] Building managers are often the only people who understand the primitive metadata, and are not adept at writing regular expression programs to transform the primitive metadata to a common and more understandable namespace themselves

[2] Transformation of about 70% of the primitive sensor metadata of these buildings, comprising 1600 and 2600 sensors, took only 21 and 27 examples respectively

BuildSys'14, November 5–6, 2014, Memphis, TN, USA.
ACM 978-1-4503-3144-9.
http://dx.doi.org/10.1145/2674061.2675031



(a) Our synthesis technique learns from an expert-provided example, applies it on the existing primitive sensor metadata to the extent it can, and requests a new example to help transform the remaining primitive metadata.

(b) Instance of an expert-provided metadata transformation example and automated metadata transformation from the rules learned from that example. Note that our program synthesis technique does not yet know how to transform the metadata string `ARS` to a metadata tag in the common namespace. Hence it needs another example from the expert.

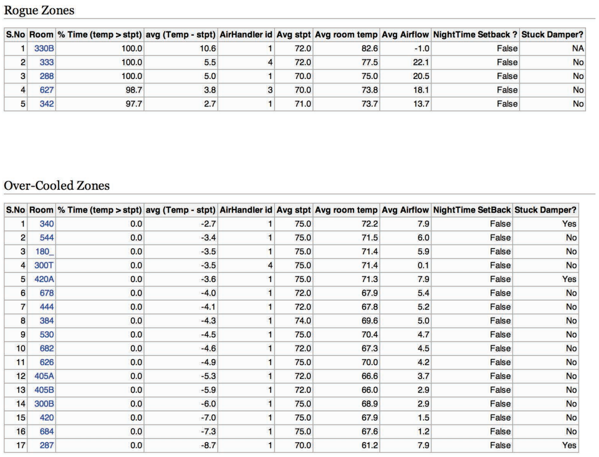
Figure 1: Illustration of the process of transforming primitive sensor metadata of a building, and thereby obtaining *semantic relationships* between the sensors. Note that the example provided by the expert not only denotes the location and type of sensor `BLDA1R435__ART`, it yields the relationship between the sensor and a particular air handling subsystem.

temp setpoint having the same value for **room-id**, and check whether the temperature is always more (or conversely less) than its respective setpoint (factoring in a tolerance factor). Figure 2, shows a list of zones which were always hot (listed under *Rogue Zones*) and a list of zones which were always cold (listed under*Over-Cooled Zones*) for a particular building in our testbed. Three of the hot zones were served by `Air` Handler 1 (the fifth column), which resulted in a lot of over-cooled zones under the same air handler. This automated analysis identified that fixing the three hot zones under `Air` Handler 1 would not only lead to a reduction in energy consumption, but also improve comfort for a majority of the over-cooled zones.

Identifying stuck dampers: A stuck air flow damper results in uncomfortable zones because the dampers do not moderate the amount of cold air flowing into a zone. Automatically identifying stuck dampers obviates the need for a technician to manually inspect air flows at each physical damper or vent location. This analysis requires on searching for the tags **room air flow damper** and checking if their data values remain unchanged. Figure 2 (the last column) shows that three of the over-cooled zones had stuck dampers.

Nighttime setback: To save energy, buildings may opt for more conservative temperature and airflow setpoints during non-office hours and weekends. Absence of nighttime setbacks helps identify a simple way to reduce building energy consumption. This analysis searches for the desired setpoints , e.g tag **room temp setpoint** and verifies whether or not they had reported different data values for office and non-office hours. Figure 2 shows that none of the rooms in our example building had night-time setbacks. In fact, only one building in our entire testbed had nighttime setbacks enabled.

We are currently working on a wider set of scalable building efficiency applications, and evaluating the robustness and scalability of our approach on a wider set of buildings.



Rogue Zones

S.No	Room	% Time (temp > stpt)	avg (Temp - stpt)	AirHandler id	Avg stpt	Avg room temp	Avg Airflow	NightTime Setback ?	Stuck Damper?
1	330B	100.0	10.6	1	72.0	82.6	-1.0	False	NA
2	333	100.0	5.5	4	72.0	77.5	22.1	False	No
3	288	100.0	5.0	1	70.0	75.0	20.5	False	No
4	627	98.7	3.8	3	70.0	73.8	18.1	False	No
5	342	97.7	2.7	1	71.0	73.7	13.7	False	No

Over-Cooled Zones

S.No	Room	% Time (temp > stpt)	avg (Temp - stpt)	AirHandler id	Avg stpt	Avg room temp	Avg Airflow	NightTime SetBack	Stuck Damper?
1	340	0.0	-2.7	1	75.0	72.2	7.9	False	Yes
2	544	0.0	-3.4	1	75.0	71.5	6.0	False	No
3	180_	0.0	-3.5	1	75.0	71.4	5.9	False	No
4	300T	0.0	-3.5	4	75.0	71.4	0.1	False	No
5	420A	0.0	-3.6	1	75.0	71.3	7.9	False	Yes
6	678	0.0	-4.0	1	72.0	67.9	5.4	False	No
7	444	0.0	-4.1	1	72.0	67.8	5.2	False	No
8	384	0.0	-4.3	1	74.0	69.6	5.0	False	No
9	530	0.0	-4.5	1	75.0	70.4	4.7	False	No
10	682	0.0	-4.6	1	72.0	67.3	4.5	False	No
11	626	0.0	-4.9	1	75.0	70.0	4.2	False	No
12	405A	0.0	-5.3	1	72.0	66.6	3.7	False	No
13	405B	0.0	-5.9	1	72.0	66.0	2.9	False	No
14	300B	0.0	-6.0	1	75.0	68.9	2.9	False	No
15	420	0.0	-7.0	1	75.0	67.9	1.5	False	No
16	684	0.0	-7.3	1	75.0	67.6	1.2	False	No
17	287	0.0	-8.7	1	70.0	61.2	7.9	False	Yes

Figure 2: A web report generated through a scalable building efficiency application on one building in the University of California, Berkley campus.

2 References

[1] Project haystack. `http://project-haystack.org/`.

[2] A. Bhattacharya, D. E. Culler, J. Ortiz, D. Hong, and K. Whitehouse. Enabling portable building applications through automated metadata transformation. Technical Report UCB/EECS-2014-159, EECS Department, University of California, Berkeley, Aug 2014.