Towards Automatic Context Inference for Sensors in Commercial Buildings

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Computer Science)

by

Dezhi Hong

August 2018

© 2018 Dezhi Hong

Abstract

Commercial and industrial buildings account for a considerable fraction of all the energy consumed in the U.S., and reducing this energy consumption has become a national grand challenge. Based on the large-scale deployment of sensors in modern commercial buildings, many organizations are applying data analytics to the thousands of sensing and control points to detect wasteful, incorrect and inefficient operations for energy savings. Scaling this approach is challenging, however, because the metadata about these sensing and control points is inconsistent between buildings, or even missing altogether. As a result, an analytics engine cannot be applied to a new building without first addressing the issue of *mapping*: creating a match between the sensor stream context and the inputs of a data analytic engine. This mapping process currently requires significant integration effort and anecdotally can take a week or longer for each building. Thus, metadata mapping is a major obstacle to scaling up building analytics.

The overarching goal of this research is to enable automatic inference of the sensor context such as its type, location, and relationships to others, so that building analytics can be quickly applied at scale. Bearing this goal in mind, we have developed a suite of techniques to infer the sensor context (i.e., the metadata), requiring *minimal* human intervention. At the core are *fully automated* techniques that infer two kinds of contextual information about each sensor: the type and its relationship with other sensors. The type inference technique leverages information from existing well mapped buildings to help the inference for a new building, while the relationship inference technique builds upon the intuition that connected equipment or co-located sensors are exposed to the same real world events, thus exhibiting correlated changes in their data. We have also formulated methods to complement the automatic type inference technique in the absence of already mapped buildings or when faced with high-dimensional data. The techniques proposed in this dissertation represent a first step towards technology that would enable any new building analytics to scale quickly to the 10's of millions of commercial buildings across the globe, with the minimal need of manual mapping on a per-building basis. With the advent of Internet of Things (IoT) and proliferation of sensory data, continuously inferring context for the data at scale will become inevitable, and we believe the techniques presented in this dissertation are generally applicable to the broader picture of IoT.

Approval Sheet

This Dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

Dezhi Hong

This Dissertation has been read and approved by the Examining Committee:

Kamin Whitehouse, Adviser

John A. Stankovic, Committee Chair

Hongning Wang

John Lach

Quanquan Gu

David E. Culler

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

August 2018

Acknowledgments

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview and Motivation

People spend 93% percent of their life in buildings [2], and according to recent reports from the U.S. Department of Energy [3, 4], commercial and industrial buildings in the U.S. account for almost 20 percent of the country's total energy use and a good 30 percent of that energy is used "inefficiently or unnecessarily." Reducing this energy usage is a national grand challenge: in 2011, the U.S. government launched the Better Buildings Challenge to make these buildings at least 20 percent more efficient by 2020 [5]. To achieve this goal, many organizations are applying data analytics to the thousands of sensing and control points in a typical commercial building to detect wasteful, incorrect and inefficient operations, and to optimize for occupants' comfort. A building data analytics engine, such as thermal comfort optimization, fault detection and diagnosis, and model predictive control [6, 7, 8, 9], typically connects to and pull data from the points in a building, in order to monitor and assess the operations of the building. For example, an analytical tool that detects over-heated rooms needs to locate a room and its associated points such as temperature measurement and room temperature setpoint.

Despite the promise of building analytics, its roll-out significantly lags behind and is adopted in far less than half of the buildings nationwide [10]. This is because of the challenge of connecting sensor data to the context in which it was generated, in a normalized and semantically consistent way. In particular, these analytics engines are tightly coupled to the database schemata and metadata[1] conventions in the buildings for which they were designed and thus cannot easily be applied to different buildings where the type, location, and relationships between sensors are represented differently. As a result, an analytics engine cannot be applied to a new building without first addressing the issue of *mapping*: creating a match between the sensor streams and

[1]We shall note that the terms "context" and "metadata" are used interchangeably in this dissertation.

the inputs of a data analytics engine. In practice, the process of mapping a new building to the inputs of an analytics engine is currently a manual process that often involves a technician visiting the building to visually inspect the sensor and equipment installation. The mapping process requires significant integration effort and anecdotally can take a week or longer for each commercial building. For large organizations that are applying this approach to hundreds or more buildings, such as Microsoft's 88 Acres project [11], this process can take years. Thus, sensor metadata mapping is a major obstacle to applying building analytics at scale.

Even if this highly manual process is performed once, the need for additional mapping is not necessarily eliminated. New types of metadata will be required as the building is modified or renovated, as the equipment is upgraded, or as new conditions and algorithms are added to the analytics engine. Additionally, the mapping problem cannot be solved simply by investing more person hours into the problem. Even after a building is fully mapped, a new analytics engine may be developed that requires a different kind of metadata, e.g., which devices are on the northern side of the building, or which sensors are in the same HAVC zone. These and other types of metadata may even never have been encoded in the original databases at all. Thus, as energy models and building analytics engines become more nuanced, the mapping problem will become increasingly important. As a result, we envision a system that will allow an advanced analytical engine to quickly connect to and analyze the data from a commercial building. It would extract or infer metadata values about the sensing and control points, and map them to a normalized standard metadata schema. The system should also enable any new building analytics engine to quickly be applied to the 10's of millions of commercial buildings across the globe. Doing so would enable a new market where boutique analytics could quickly be matched with the buildings they would benefit the most.

Several solutions have been proposed recently to facilitate the mapping problem, inferring various aspects of sensors, including the type, location, and relations between them. Schumann et al. [12] develop a probabilistic framework to classify sensor types based on the similarity between a point name and the entries in a manually constructed dictionary. However, the dictionary needs to be created on a per building basis, hence still time-consuming and unscalable. Bhattarcharya et al. [136] formulate a solution that derives a set of regular expressions from a handful of labeled examples to extract the sensor context, where the example selection is aided by time series data feature-based clustering. However, they assume a small, limited amount of patterns in the naming convention for a building, which might not always be true. Marco et al. infers [13] the relation between VAVs and AHUs by perturbing the operation of each AHU and observing the response of VAVs. However, this approach takes weeks to execute and requires knowledge of when and how to perturb operations in a way that does not interfere with building needs. These approaches can significantly reduce the time required for mapping in each new building, but they all still require some manual effort, one building at a time. We would still need a framework that can be (almost) automatically executed.

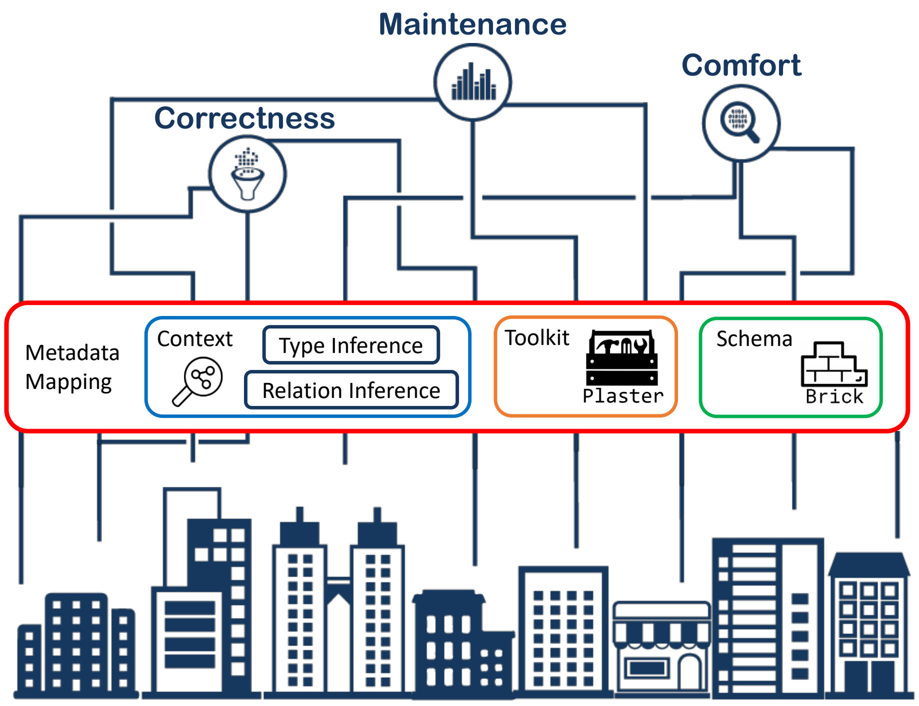


Figure 1.1: Automatic Metadata Mapping is a first step towards scalable building analytics, which involves 1) inferring the context of sensors with an easy-to-use tool and 2) expressing it following a standardized schema. This dissertation is dedicated to the first perspective.

The overarching goal of this research is to enable inference of sensor contextual information requiring *minimal* human intervention. To address this need, we have designed and built *Data Holmes*. The basic idea is that a user such as building manager only needs to provide the point names and time series data of sensors, together with the labels for only a few example sensors, as input to our framework, and Data Holmes will then automatically infer the contextual information about the sensors in the building. Data Holmes infers two key kinds of information — the type of each sensor and the relations between sensors. Buildings share similarity with regard to the types of sensors installed, occupancy patterns, operation schedules, etc, so when inferring the type for each sensor in one building, Data Holmes leverages the information from sensors in other well mapped buildings and completely automatically performs the inference; it will require a few human inputs, only if there are no mapped buildings available otherwise. When Data Holmes infers the relations between equipment (e.g., the functional connections), the process is fully automatic where it first detects the "events" in the time series data for each sensor and then performs correlation analysis to identify the relations. The intuition is that connected equipment or co-located sensors are exposed to the same real-world events, thus exhibiting correlated changes in their time series measurements.

1.2 Hypothesis Statement

In this dissertation, we argue that it is possible to infer the contextual information (including the type and relation) about sensors in a building with minimal manual input, using their point names and time series data, so that data analytics can be quickly applied at scale.

1.3 Technical Challenges

This dissertation addresses a number of key technical challenges in building a framework that automatically infers sensor context and in facilitating this line of research in the future.

First, no two buildings are identical. When leveraging existing mapped building(s) to help infer the type for sensors in a new building, given the difference across buildings, to accurately quantify the underlying similarity in sensors between the mapped and new buildings and subsequently effectively apply the information is therefore challenging.

Second, human input is prohibitive. When there are no other mapped buildings available to assist the type inference in a new building, it is necessary to ask a human expert to provide labels for example sensors; and typically labeling one sensor can cost more than one hundred dollars. A challenge is therefore how to effectively select a minimal set of example sensors for a human to label.

Third, the type inference problem space can be high-dimensional. The proliferation of cheap, ubiquitous sensing infrastructure could pose a high-dimensional computational problem — the number of sensors can significantly outnumber the readings for each sensor time series. This essentially results in more variables to estimate than observations available for each variable.

Fourth, the search space of relation inference between sensors is huge. Given a set of sensors in a building, the total number of possible pairwise relations among all sensors grows quadratically with the number of sensors, while the number of true relations is only linear in the number of sensors; identifying true relations of interest is to find a needle in a haystack.

Fifth, existing metadata inference algorithms require a benchmark. Considering the heterogeneity in the data models, learning scope, input/output format of existing metadata inference algorithms, together with the absence of a common dataset, to conduct systematic and fair comparisons between these algorithms is another challenge.

1.4 Contributions

Recognizing these challenges, we make the following contributions in this dissertation:

- We design a transfer learning-based method to *fully automatically* infer the type for points in one building, leveraging any existing already mapped buildings; on a new building, the proposed solution can automatically label at least 36% of the points with more than 85% accuracy, and in some cases up to 81% of the points with 96% accuracy.
- We formulate a novel, effective yet general active learning approach to perform semi-automated type classification requiring *minimal* human labeling input, by exploiting the commonalities among sensor names; our solution achieved more than 92% accuracy with at least 50% fewer examples than the state-of-the-art active learning algorithms. This semi-automated algorithm would complement the first algorithm when there are no existing mapped buildings available.
- We propose a *new measure* to quantify the similarity between time series, based on which we perform clustering to identify clusters of the same type of sensors, in the *high-dimensional* setting — the number of sensor time series is more than the number of readings per sensor; the method can achieve more than 80% clustering accuracy on real-world data, which is 20% higher than the state-of-art baselines. We also conduct theoretical analysis for the algorithm to prove when it is guaranteed to succeed.
- We develop a two-step approach to *automatically* identify the functional relationships between AHUs and VAVs, which first locates the events in the time series sequence from each equipment and then performs correlation analysis to find the connections; the algorithm achieves more than 94% accuracy on average, which is more than 10% better than the best baselines. We also briefly demonstrate how the same principle of *locating correlated events* can help identify co-located sensors.
- We design and implement an *open-sourced* integration platform comprised of unified dataset and standard benchmarks for systematically comparing and combining existing state-of-the-art metadata inference algorithms, *for the first time*. The platform provides easy-to-use programming interfaces for non-computer science practitioners such as building facilities technicians to compose their own workflows, in order to develop and evaluate (new) metadata inference algorithms.

1.5 Organization of the Dissertation

In the rest of the dissertation, we will first overview the related works to this research (Ch. 2), explain multiple techniques on inferring the type for sensors (Ch. 3 - Ch. 5), introduce a method for relationship inference

(Ch. 6) as well as a framework for users to perform these tasks (Ch. 7), and finally conclude the dissertation (Ch. 8). In particular,

- Chapter 2 presents the state-of-the-art in various related topics, including the problem of sensor metadata inference, transfer learning, active learning, high-dimensional clustering, and event detection, that are involved in this dissertation.
- Chapter 3 describes the design and evaluation of Building Adapter — a transfer learning-based algorithm that leverages existing well mapped buildings to assist the type inference for a new building.
- Chapter 4 provides in-depth explanation and evaluation of the active learning-based method, which requires a minimal amount of human labeling input to efficiently learn the sensor type, as a complement to Building Adapter.
- Chapter 5 elaborates the motivation, algorithm, and evaluation of the high-dimensional clustering method for recognizing sensor type clusters.
- Chapter 6 presents the solution to automatically infer the functional relationships between equipment (i.e., which AHU is connected to which VAVs), using the sensory time series measurements from each equipment.
- Chapter 7 describes the details of an open-sourced integration platform for systematically benchmarking and combining existing metadata inference algorithms.
- Chapter 8 concludes the dissertation by summarizing the contributions and discussing future improvements as well as new directions.

Chapter 2

Background and Related Work

This chapter first explicates the "metadata challenge" in commercial buildings and then presents the state-of-the-art in topics related to sensor metadata inference, transfer learning, active learning, high-dimensional clustering, and event detection that are involved in this dissertation.

2.1 Metadata Heterogeneity

In modern commercial buildings, a sensing or control "point" is a sensor, a controller, or a software value, e.g., a temperature sensor installed in an office room, or a room temperature setpoint. The metadata about the point indicates the physical location, the type of sensor or controller, how the sensor or controller relates to the mechanical systems, and other important contextual information. Most of the time, the metadata, often also referred to as *point names*, is encoded as short text strings comprised of an arbitrary number of concatenated abbreviations. Table 2.1 lists a few point names of sensors in three different building management systems (Trane[1], Siemens[2] and Barrington Controls[3]). For example, the point name `SODA1R300__ART` is constructed as a concatenation of the building name (`SOD`), the air handling unit identifier (`A1`), the room number (`R300`), and the sensor type (`ART`, area room temperature). As the name indicates, this point measures the temperature in a particular room; and it also indicates the air handling unit that can affect the temperature in this room. These point names are often esoteric and meaningful to only domain experts such as building managers and facilities technicians. Furthermore, clearly distinct naming conventions — generally devised by the equipment, vendor, and manufacturer — are used in these buildings. For example, the notion of *room temperature* is encoded with a different abbreviation in each of the three buildings as: `Temp`, `RMT`, and `ART`, respectively.

[1]`http://www.trane.com/`
[2]`http://www.siemens.com/`
[3]The company is no longer in business.

Such variations across different buildings impose great difficulty in quickly interpreting the data streams and deploying automated analytic solutions.

Building	Point Name
A	`Zone Temp 2 RMI204`
	`spaceTemperature 1st Floor Area1`
B	`SDH_SF1_R282_RMT`
	`SDH_S1-01_ROOM_TEMP`
C	`SODA1R300__ART`
	`SODA1R410B_ART`

Table 2.1: Example point names for temperature sensors from three different buildings.

2.2 Related Work

2.2.1 Metadata Inference

There have been recent efforts in addressing the sensor metadata mapping problem studied in this dissertation. Dawson-Haggerty et al. [14] and Krioukov et al. [15] introduced a Building Operating System Service stack, whereby the underlying building sensor stock is presented to applications through a driver-based model and an application stack provides a fuzzy-query based interface to the namespace exposed through the driver interface. Although this architecture has some useful properties for easing generalizability across buildings, the driver registration process is still performed manually. Schumann et al. [12] developed a probabilistic framework to classify sensor types based on the similarity of a raw point name to the entries in a manually constructed dictionary. However, the performance of this method is limited by the coverage and diversity of entries listed in the dictionary, and the dictionary size becomes intractable when there exist considerable variations in the names of the same sensor type, or conflicting definitions of a dictionary entry in different buildings. Bhattarcharya et al. [136] exploited a programming language based solution that derives a set of regular expressions from a handful of labeled examples to extract the contextual information about sensors from their point names. This approach assumes a (reasonably) consistent format for the point names in a building, which might not be true in practice (as shown in Table 2.1), and the human input requires expertise in regular expression. Balaji et al. [16] proposed a method that first clusters point names and then acquires the labels for a few examples from each cluster to learn a classifier for predicting the type of points. They assume points within each cluster to have the same type, which is often not true and therefore introduces significantly more labeling burden on the human labeler. Gao et al. [17] comprehensively studied how the time series data features can be exploited to learn sensor types with various classical supervised algorithms. While their focus is more concerned with the effectiveness of different algorithms, they do not specifically aim to minimize the number of labels, i.e., the amount of human input.

Among the first papers attempting to automatically infer the relations between different sensors, Smith et al. [18] used linear models to infer the relationship between temperature sensors and heating vents, and Hong et al. [19] used a correlation based metric over time series to infer which sensors are in the same room. Koc et al. [20] later also used a correlation-based method to infer spatial relationships between discharge air and zone temperature sensors from different rooms. For the method in Chapter 6, we focus on a different type of relationship — the functional connection between equipment, and so all these approaches do not directly apply. Additionally, these approaches assume a known structure of building design or building equipment, whereas our solution is expected to be building-independent. A recent work [13] is focused on the same type of relation as in Chapter 6, where the solution relies on manually turning off each AHU and leveraging the responses in VAVs to identify the connections. This approach requires knowledge about when and how to perturb the operations of the equipment in a way that does not interfere with the normal building needs, and furthermore, it takes weeks to execute. However, the approach still contains significant errors and must be applied to each individual building manually, while the goal of our work is to enable inference on new buildings with higher accuracy, yet with *minimal* to *almost no* manual intervention.

Besides, there are three major bodies of related work to the methods studied in this dissertation, viz, 1) schema matching in database, 2) transfer learning, active learning, and high-dimensional clustering in machine learning, and 3) event detection in data mining.

2.2.2 Schema Matching

Automatic schema matching [21] is a classical problem in the database community where correspondences between elements of two schemas are identified as part of the data integration process. Many techniques have been proposed to achieve partial automation of the match operation for specific application domains. Doan et al. [22] asked a user to provide semantic mappings for a small set of data sources and then train a set of learners with existing machine learning approaches to find the mappings for new data sources. Dhamankar et al. extended [22] to a semi-supervised setting, where domain-specific knowledge is introduced for complex expressions learning [23]. Madhavan et al. [24] exploited a large collection of schemata with known mappings to learn a prior distribution of the elements and their properties. The learned prior distribution is then used as constraints to help a suite of base learners to complete the matching. Though adapting learning techniques, these works mostly focus on the offline supervised settings and do not emphasize the efficiency of learning methods, i.e., the amount of human input required.

2.2.3 Transfer Learning

In our context, applying transfer learning across buildings, e.g., exploiting the labels in already well annotated building(s), for sensor type classification can save extra effort in manual annotations. There are several categories of transfer learning, e.g., inductive, transductive, and multi-task transfer learning as comprehensively surveyed in [25]. Inductive transfer learning [26] assumes the set of class labels in the target domain is different from those in the source domain, and aims at achieving high classification performance in the target domain by transferring knowledge from the source domain. Multi-task transfer learning [27] has a similar setting, but tries to learn from the target and source domains simultaneously. Transductive transfer learning [28] assumes the source and target domains have the same set of labels, but different marginal distribution of features or conditional distribution of labels. This breaks the basic identical and independent assumption in classical supervised learning models and makes them inept. Typical solutions in transductive transfer learning reweigh the source domain trained classifiers' predictions in target domain, e.g., instance-based local weighting [29, 30, 31]. But these solutions usually assume that only the marginal distribution of features differ in the source and target domain. Ensemble methods are therefore explored to assign different weights to a set of classifiers to accommodate the varying conditional probabilities of labels in the target domain [32, 33]. Our problem setting falls into this category: we assume we have well-labeled instances in a source building, but do not have any labeled instances in the target building. When designing the *Building Adapter*, we exploit different properties of a sensor point to perform the transfer learning: sensor's timeseries data is utilized to estimate a diverse set of classifiers to transfer knowledge from the source building to the target building, while sensor names in the target building are used to compute the ensemble weight of classifiers during knowledge transfer. We will defer the details to Chapter 3.

2.2.4 Active Learning

The algorithm detailed in Chapter 4 builds upon active learning to reduce the amount of human labeling required, as a complement to *Building Adapter* when no mapped buildings are available. The main idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer labeled training instances if it is allowed to choose the data from which it learns [34]. Therefore, the key research question in active learning is evaluating the informativeness of unlabeled instances for querying. Various solutions have been proposed from different perspectives, including uncertainty reduction [35], query by committee [36], and expected error reduction [37]. In particular, Nguyen and Smeulders also incorporated the idea of clustering into active learning, as we do in our proposed algorithm, to select the most informative instances [38]. Their proposed query strategy gives priority to instances that are close to both classification boundary and cluster centroid. Dasgupta and Hsu utilized a hierarchical clustering structure to alleviate sampling bias and improve learning efficiency [39]. They present an algorithm that is statistically consistent and guarantees to have better label complexity than supervised learning. However, in these two methods clustering is performed in an ad-hoc manner, e.g., with predefined cluster size; and neither of them considers the label proximity between adjacent instances or propagates labels to reduce the amount of labels required for model training. Our proposed active learning algorithm effectively leverages a clustering step to identify examples that are both *informative* and *representative* for training a classifier, and also employs a label propagation mechanism to further reduce the human labeling effort.

There has been a substantial body of work on time series clustering, and we next briefly overview two related categories to our method in Chapter 5: clustering based on similarity and subspace clustering.

2.2.5 Similarity/Distance-based Time Series Clustering

A wide range of classical similarity/distance metrics have been developed and studied [40], including Pearson's correlation coefficient [41], cosine similarity [42], autocorrelation [43], dynamic time warping [44, 45], Euclidean Distance [46], edit distance [47], distance metric learning [48, 49], and so on. Studies have also shown that time series can be modeled as generated from Hidden Markov Models [50, 51], and the estimated weight for each mixture can be used to cluster the time series. Recently, Ryabko et al. [52] considered brain-computer interface data for which independence assumptions do not hold, and for clustering they proposed a new distance metric to measure the similarity between two time series distributions. Khaleghi et al. [53] formulated a novel metric to quantify the distance between time series and proved the consistency of k-means for clustering processes according only to their distributions. However, in the aforementioned studies, they either did not provide theoretical analysis of the performance or only handled settings where the number of time series d is smaller than the number of observations T. Different from the above similarity or distance metrics, we define the similarity between time series from a new perspective — time series are clustered based on how much they can be predicted by each other.

2.2.6 Subspace Clustering (SC)

Another relevant line of research on high-dimensional data analysis is subspace clustering [54], where the assumption is that data lie on the union of multiple lower-dimensional linear spaces and data points can be clustered into the subspace they belong to. SC has been widely applied to face images clustering [55], social graphs [56] and so on. Recently, extensions to handling noisy data [57, 58, 59] and data with irrelevant features [60] have been studied as well. SC achieves the state-of-art performance while enjoying rigorous theoretical guarantees. The key difference between SC and our method is two-fold: first, SC assumes data lie on different subspaces and even data in the same subspace are independent and identically distributed (i.i.d.), while we assume the time series follow a Vector Autoregression model and are dependent for the ones in the same cluster; second, SC mathematically solves for each data point a linear regression problem with all other data points being the candidate, while in contrast, our study solves the regression problem to estimate the prediction weights between observations from different time stamps using all the time series.

In Chapter 6, we will present a method to identify the functional relations between AHUs and VAVs, which first locates the latent events in the time series data and then performs correlation analysis based on the events. We next discuss about the line of work on event detection as well as a few other related ones.

2.2.7 Change and Event Detection

Change detection for time series [61, 62, 63] locates the changing point of data distribution, and has been widely applied to various domains, including medical condition monitoring [64], climate change [65], image analysis [66], etc. Particularly, change detection has been extended to detect events, in supervised manner (decision tree [67], support vector machines [68], and Bayesian net [69]), and unsupervised manner (Gaussian Mixtures [70], Kalman Filter [71], Poisson Process [72], and Scan Statistics [73]). Our method falls into the unsupervised category. However, events in our context are real-world activities (e.g., fans turning on) that can just have the same values as non-events; and for this reason, the aforementioned unsupervised methods cannot directly apply to our context. Instead, to identify the events in our context, we seek to learn a noise model and a transition model to differentiate different operating states of equipment. In addition, compared with standard algorithms that model kinematics of the data, such as Kalman Filter, our model learns the state parameters by holistically using all the data, and the learned model parameters are globally shared to all the observations in time. Furthermore, while the above methods focus on domain-specific event detection, our approach does not require any domain-specific knowledge about the data, and so we expect it to be more generally applicable.

Event detection also shares similarity with novelty detection [74], outlier [75] and anomaly detection [76], frequent and periodic pattern mining [77]. Outlier and anomaly detection is focused more on identifying nonconforming or irregular patterns in the data, while events typically affect a subgroup of the data and they are not necessarily individual outliers. Frequent and periodical pattern mining search for periodic and frequent events in the data, whereas our model is able to identify acyclic and rare events.

At first glance, detecting events in time series appears to be related to time series segmentation [78], subsequence clustering [79], or the combination of both [80, 81]. However, detecting events in a time series is more difficult than segmentation, since events can happen in different manners and can possibly span over multiple segmentations. For subsequence clustering, it tends to assign nearby points into the same cluster, which could unnecessarily introduce false positive events. Additionally, in our context, detecting events in each time series is only a first step and subsequently based on the events, it requires solving a more challenging matching problem where we seek to find the one-to-many mapping between an AHU and a group of VAVs of unknown size. The existence of groups of time series with unknown sizes a priori makes the recent advances in simultaneous segmentation and clustering [82], which requires the knowledge about the number of sensors in a group beforehand, inapplicable to our problem.

Chapter 3

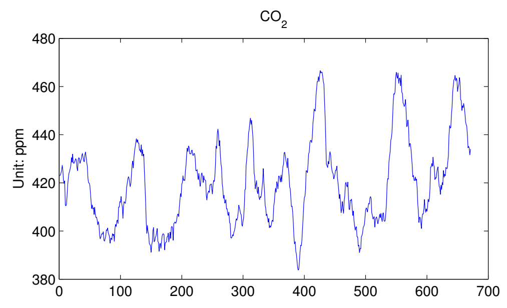
Fully Automated Type Classification

This chapter presents the design and implementation details of *Building Adapter*, a *fully automated* approach to inferring the type for sensors, which is a sheer distinction from previous works that still require human input [12, 136, 16, 17]. It is a first step towards scalable building analytics which will require new techniques to automatically infer the sensor contextual information such as its type in a building without manual labeling. The technique discussed in this chapter is based on the observation that we often have access to a few fully mapped buildings, and we can *transfer* the information from these buildings to assist the inference in a new building. The key insight that guides our solution is that a sensor typically has two attributes — its recorded name (a text string) and its numerical readings generated over time. In particular, the sensor names are likely to be good indicators of metadata structure since people would usually follow a certain naming convention, but the conventions might not be consistent across buildings, as shown in Table 2.1. While the sensor readings are more consistent across buildings but are likely to be poor indicators of sensor types. For instance, room temperature will be between 60 and 70 Fahrenheit degrees, no matter in which building. *Building Adapter* combines the complementary strengths of these two attributes to recognize the type of points in a building.

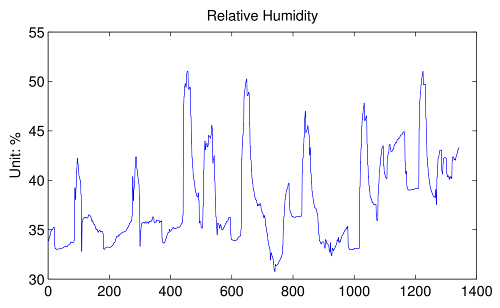
More specifically, our approach comprises three steps: 1) we start with a fully labeled building and train multiple statistical classifiers based on the sensor stream data. These classifiers predict sensor types based on the raw data, e.g., readings from air temperature sensors are different and change more slowly than those generated by light or CO_2 sensors; 2) in an unlabeled building, we cluster the points based on patterns in their names such as the phrases "temp" or "occ"; 3) we ensemble the classifiers from the first building to predict for the points in the second building, and assign higher weights to classifiers whose predictions on an instance's neighborhood in the target building are more consistent with the name feature defined clusters. In the rest of this chapter, we first explain a few essential primers to understanding the approach and then describe the details of each step.

3.1 Feature Representation

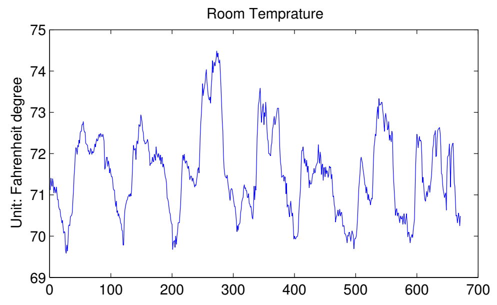
Building Adapter exploits two common attributes of the sensor points in a building — the actual sensor reading data and the text string-based point names, both of which play an important role for differentiating sensor types. In this section, we describe the construction of two different sets of features, i.e., the data features and the name features.



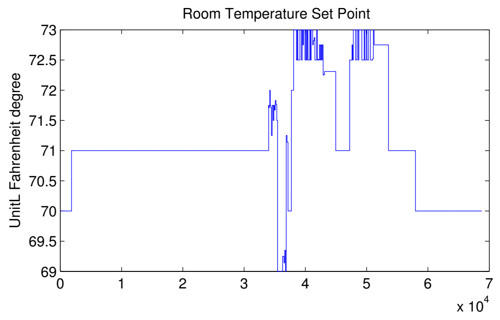
(a) CO_2



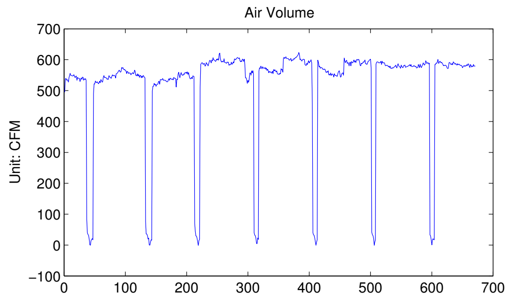
(b) Humidity



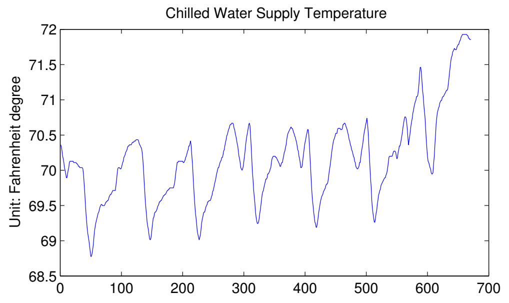
(c) Room Temperature



(d) Room Temperature Set Point



(e) VAV Air Volume



(f) Chilled Water Supply Temperature

Figure 3.1: Different types of sensor generally have different amplitudes that can be separated and binned to characterize the data.

3.1.1 Data Features

A signal[1] in the time domain is a trend of sensor reading. Different types of sensors generally have different amplitudes that can be separated and binned, as shown in Figure 3.1. Similar to piecewise aggregated approximation (PAA [83]) — where the mean value is calculated in a fixed-length window — we compress the signal by computing a set of summary statistics over fixed-length windows. Table 3.1 shows a summary of the statistics we calculate as data features.

Category	Statistical Function	Acronym
Extrema	Minimum	min
	Maximum	max
Average	Median	emd
	Root Mean Square	rms
Quartiles	1st and 3rd Quartiles	1q, 3q
	Inter-quartile range	iqr
Moments	Variance	var
	Skewness	skew
	Kurtosis	kurt
Shape	Linear Regression Slope	slope

Table 3.1: Statistical features extracted in window level for each time series data.

Our feature extraction process consists of three steps. First, each sensor trace is segmented into N hour-long windows with 50% overlap between consecutive windows. Second, for each time window, we compute the statistics shown in Table 3.1. For example, the vector for MIN is computed as follows: $MIN = \{min^1, min^2, ..., min^N\}$, where N is the number of time windows. We compute a similar vector for each statistic shown in the table. Third, we compute a statistical summary of these vectors. For each vector we compute the *minimum*, *maximum*, *median*, and *variance*, resulting in a feature vector containing 44 variables:

$$\begin{aligned} x^D = \{&min(MIN), max(MIN), \\ &median(MIN), var(MIN), \\ &\ldots \\ &min(SLOPE), max(SLOPE), \\ &median(SLOPE), var(SLOPE)\}, \end{aligned}$$

and x^D is the data feature vector for each sensor stream used in our study.

[1]In this chapter, we use the term “signal”, “trace” and “time series” *interchangeably*.

3.1.2 Name Features

Sensor point names are short text strings comprised of several concatenated abbreviations, as shown in Table 2.1. To extract features from a point name, we first convert all alphabetical characters to lower cases and trim out numerical characters. For example, `Zone Temp 2 RMI204` becomes {`zone, temp, rmi`}. Next, we compute k-mers [84], which includes all the substrings of length k and helps measure sequence similarity without requiring alignment, to capture variations in type abbreviations. For example, the type "temperature" might be encoded as "tmp" or "temp" and only considering the bag-of-words [85] will miss the similarity within a word boundary, leaving these two strings far in the vector space after transformation. Therefore, we adopt k-mers, which refers to all the possible substrings of length k in a given string. This feature representation is widely used in protein and gene sequence analysis [86, 87] and we limit our k-mers computation within a word boundary. For example, for a string "ABCDE", the 3-mers (k=3) generated would be {`ABC, BCD, CDE`}.

In general, a smaller k will increase the overlapping between the generated k-mers, making points less differentiable. Therefore, we compute all k-mers of length 3 and 4 for each point name. For example, {`zone, temp, rmi`} yields a set of k-mers {`zon, one, tem, emp, rmi`} when k=3. A dictionary of k-mers is constructed with all the k-mers generated from each point name. Each point name is transformed into a feature vector based on the frequency of k-mers in it. For example, a set of k-mers {`zon, tem, emp, zon`} will be transformed to a vector `(2,0,1,1,0)` with the dictionary {`zon, one, tem, emp, rmi`}, meaning `zon` occurs twice, `one` does not appear, and so forth. This feature representation of point names will be used for clustering on the new building.

3.2 Non-parametric Bayesian Clustering

We shall note that in both of the proposed techniques in this chapter and the next chapter, name feature distribution $p(x^P)$ is exploited via its latent clustering structure. We choose Gaussian Mixture Model (GMM) [88], a partitional clustering algorithm, to perform the clustering.

In GMM, the cluster label for every instance is treated as a latent variable, which is drawn from a multinomial distribution $p(c)$, i.e., $p(c) \propto \alpha_c$, where $\forall c, \alpha_c \geq 0$ and $\sum_c \alpha_c = 1$. In any given cluster c, the conditional data likelihood of an instance x is specified by a multivariate Gaussian distribution. To reduce the number of parameters to be estimated, we choose the isotropic Gaussian in our solution,

$$p(x^P|c) = (2\pi\sigma^2)^{-d/2} \exp - \frac{(x^P - \mu_c)^\mathsf{T}(x^P - \ mu_c)}{2\sigma^2} \tag{3.2.1}$$

where the variance σ^2 is shared by all the clusters. $\{\alpha_c, \mu_c\}_{c=1}^{k}$ and σ are considered as model parameters in GMM.

However, in GMM, we need to manually specify the number of clusters for a given input data set, and the clustering result of GMM is very sensitive to such setting. More importantly, in our solutions, for the automated technique, usually there is more than one pattern in the point names even for the same type of sensors; therefore we cannot assume a class has only one cluster. As for the semi-automated technique, we need to perform sub-clustering whenever we encounter a cluster contradicting with the current classifier's prediction. It is impossible to pre-define the size of those sub-clusters during active learning. To make clustering feasible in our solution, we appeal to a non- parametric Bayesian solution: we assume the model parameters (α, μ) in each cluster are random variables, which are drawn from a Dirichlet Process prior [89].

A Dirichlet Process $DP(G_0, \eta)$ with a base distribution G_0 and a scaling parameter η is a distribution over distributions [89]. The base distribution G_0 specifies the prior distribution of model parameters, e.g., mean parameter μ in each cluster, and the scaling parameter η specifies the concentration of samples drawn from DP, e.g., cluster proportion $p(c)$. An important property of the DP is that though the draws from a DP have countably infinite size, they are discrete with probability one, which leads to a probability distribution on partitions of data. The number of unique draws, i.e., the number of clusters, varies with regard to the data and therefore is random, instead of being pre-specified.

As a result, with the introduced $DP(G_0, \eta)$ prior, data density in a given collection of instances can be expressed using a stick-breaking representation [90]:

$$p(x^P) = \sum_{c=1}^{\infty} \alpha_c \mathcal{N}(x^P | \mu_c, \sigma) p(\mu_c | G_0) \tag{3.2.2}$$

where $\alpha = \alpha_{c=1}^{\infty} \sim Stick(\eta)$ represents the proportion of clusters in the whole collection. The stick-breaking process $Stick(\eta)$ for the cluster proportion parameter α is defined as: $\alpha'_c \sim Beta(1, \eta), \alpha_c = \alpha'_c \prod_{i=1}^{c-1}(1 - \alpha'_i)$. Since the variance σ^2 is fixed in all clusters, we use a conjugate prior for μ in G_0, i.e., for $\forall c, \mu_{ci} \sim \mathcal{N}(a, b)$, with the assumption that each dimension in μ_c is independently drawn from a univariate Gaussian. This will greatly simplify the later on inference procedure.

Because the data density distribution defined in Eq (3.2.2) only has finite support at the points of $\{\alpha_c, \mu_c\}_{c=1}^{k}$, we can calculate the posterior distribution of latent cluster labels in each unlabeled instance to discover the clustering structure for active learning. Following the sampling scheme proposed in [91], we appeal to a Gibbs sampling method to infer the posterior of cluster membership. Detailed specifications of this sampling algorithm can be found in [91]. In particular, we use the same hyper-parameter setting of (a, b) in G_0 and η for initial clustering and subsequent sub-clustering during our cluster-based active learning process.

3.3 Building Adapter — Fully Automated Type Classification

Being able to automatically map points to a normalized standard in a new building is the key to scaling up and performing widespread analysis. In this section, we will elaborate on our technique for automating the metadata mapping process for a building with the information from other well mapped buildings. Based on the insight that sensor reading data and sensor names characterize different properties of a sensor stream, we develop a transfer learning based approach to exploit features extracted from both. Specifically, we construct classifiers based on the data features because they are more likely to be consistent across buildings. However, a single supervised classifier might not perform well on all instances in the new building due to the inductive bias inherent in classifier training. Hence, we employ an ensemble of classifiers, where each classifier captures a different "perspective" in predicting the sensor type. When being applied to a new building, since different classifiers might be effective in predicting different instances, we appeal to the instance-specific local weighting method proposed in [92] to weight different classifiers while ensemble. In our solution, the weight is derived based on the consistency between a classifier's predictions and the instance's local clustering structure, which is estimated by the sensor names in the target building. In the rest of the section, we first elaborate how we construct the two different types of features and then concentrate on the proposed transfer learning method.

3.3.1 Locally Weighted Ensembles for Knowledge Transfer

To effectively encapsulate the knowledge from one labeled building and transfer it to another unlabeled one, we construct a set of different classifiers, including support vector machines [93], logistic regression [94] and random forest [95], using the same set of data features from the source building. Each classifier presents a different "perspective" for recognizing sensor types in the source building, due to the inductive bias of the underlying classifier. We refer to these classifiers as *base* classifiers in this chapter, which will be combined for type classification in the target building.

The performance of any particular classifier can vary from building to building, and different classifiers can be effective at different regions or structures in a target building: no single classifier can perform well on all instances. Therefore, to combine the knowledge in base classifiers, we employ the method from [92] to locally weigh each classifier and combine the predictions from different base classifiers in the target building. The weight is computed per classifier per instance based on the consistency between the base classifiers' predictions and the local clustering structure of the target instance. Intuitively, the estimated local weights will favor classifiers whose predictions are consistent with the estimated local structure of the target instance in the target building.

Formally, let x^D be the data feature vector of an instance x in the target building and y be its predicted class label. Given a set of k base classifiers $M_1, \ldots, M_k$ and the new testing set D_T encoded with the data features, the general Bayesian model averaging rule estimates the posterior distribution of y in x as,

$$p(y|x^D) = \sum_{i=1}^{k} p(y|x^D, D_T, M_i)p(M_i|D_T) \tag{3.3.1}$$

where $p(y|x^D, D_T, M_i) = p(y|x^D, M_i)$, because $x^D \in D_T$ and $p(y|x^D, M_i)$ is the label for x predicted by M_i and $p(M_i|D_T)$ is the probability of choosing M_i given the testing set D_T. Since $x^D \in D_T$, $p(M_i|D_T)$ is equal to $p(M_i|x^D)$, which is the locally adjusted weight for M_i, and Eq. 3.3.1 becomes,

$$p(y|x^D) = \sum_{i=1}^{k} w_{x^D}^{M_i} p(y|x^D, M_i) \tag{3.3.2}$$

where $w_x^{M_i} = p(M_i|x^D)$. A classifier M_i is expected to have a higher weight for x if M_i's predicted local structure for x is closer to its estimated neighborhood. We will next explain how the weight is calculated per instance.

3.3.2 Graph-based Weight Estimation

Ideally a larger weight should be assigned to the classifier whose predicted neighborhood structure of a testing instance in the target building is most consistent with its true neighborhood. To locally weight the classifiers in such a manner, we appeal to the technique proposed in [92]: it performs clustering in the target domain and assumes if the clustering boundary for the region where x falls, agrees with the decision boundary of M_i, a larger weight should be assigned to M_i for instance x. In other words, if the predictions made by M_i in the area surrounding x have greater consistency with the clustering results, M_i will be assigned a larger weight at x.

Based on these assumptions, the weight can be estimated with a graph-based algorithm. To compute the $w_x^{M_i}$ for M_i at instance x, we construct two neighborhood graphs: $G_M = (V, E_M)$ and $G_C = (V, E_C)$, for classification and clustering results, respectively, where each vertex is an instance in the target domain D_T. In the graph constructed based on the classifier M_i, i.e., G_{M_i}, an edge exists between two vertices (denoting the two instances are "neighbors") if and only if M_i assigns the same label for these two instances. Likewise, in G_C, an edge exists between two vertices if and only if these two instances reside in the same cluster. If the neighbors of x on both graphs significantly overlap, meaning the structure predicted by classifier is consistent with the cluster structure, M_i will then be assigned a larger weight in ensemble. So the weight $w_x^{M_i}$ for M_i at

x is proportional to the similarity of the two graphs:

$$w_x^{M_i} \propto s(G_M, G_C|x) = \frac{|V_M \cap V_C|}{|V_M \cup V_C|} \tag{3.3.3}$$

where V_M (V_C) is the set of neighbors of x on graph G_M (G_C), $|X|$ is the cardinality of set X, and $s(G_M, G_C|x)$ denotes the similarity between the two graphs. Figure 3.2 illustrates an example of neighborhood graphs for an instance x (in grey circle): classifier 1 has a similarity of 0.75 with the cluster graph while classifier 2 has a similarity of 0.5 with the cluster graph.

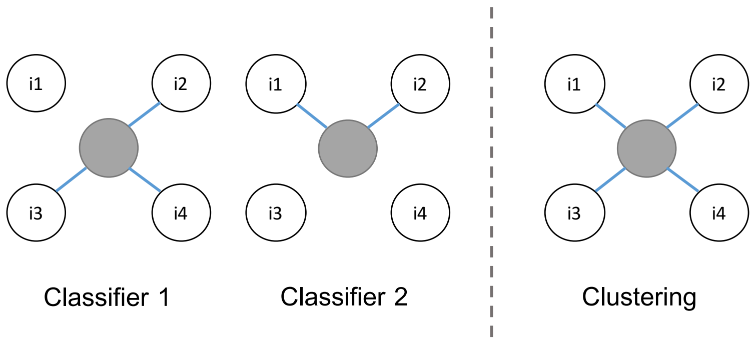


Figure 3.2: An example of local neighborhood graphs of x (in grey circle): two different classifiers predict different instances as neighbors to the target instance, where classifier 1's prediction is more similar than classifier 2's to the cluster structure.

The weight for each M_i is then calculated by normalizing among all similarity scores:

$$w_x^{M_i} = \frac{s(G_{M_i}, G_C|x)}{\sum_{i=1}^{k} s(G_{M_i}, G_C|x)} \tag{3.3.4}$$

The final prediction for x is simply $\hat{y} = argmax_y \ p(y|x^D)$ as $p(y|x^D)$ is defined in Eq. 3.3.2.

Distance-based Adjustment

In some cases the similarity score defined in Eq 3.3.4 can be problematic. Consider the case shown in Figure 3.3. The example shows two classifiers where the target instance has two neighbors. The distance between the instances is marked on the edge. Since the original similarity definition only considers the number of neighbors, both classifiers will be assigned the same weight of 0.5. However, the neighbors in classifier 1 are closer to the target instance than those in classifier 2. Therefore, classifier 1 should be assigned a larger weight. To fix this issue we include the distance between instances into consideration and adjust the similarity score as,

$$s^*(G_M, G_C|x) = 1 - \frac{\sum d_{V_I}/|V_I|}{\sum d_{V_U}/|V_U|} \tag{3.3.5}$$

where $V_I = V_M \cap V_C$, $V_U = V_M \cup V_C$, and $\sum d_{V_I}$ is the sum of distance between x to its neighbors in V_I (likewise for $\sum d_{V_U}$).

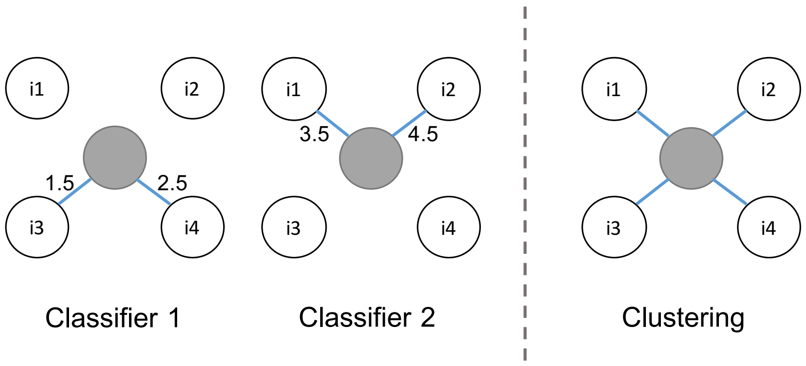


Figure 3.3: An example of local neighborhood graphs of x with distance into consideration.

Thresholding-based Adjustment

The use of a *weighted-average decision* for x among base classifiers is reasonable when at least some classifiers perform well on x. However, the similarity $s(G_{M_i}, G_C|x)$ for M_i is expected to be small when the predictions of M_i around x conflict with its true local structure. In such a case, using the decision from this classifier might be misleading. Since $s(G_{M_i}, G_C|x)$ reflects the consistency between M_i's predictions and the testing instance x's local clustering structure, we set a threshold on the average similarity score over all M_i on x to limit the usage of these base classifiers. As an adjustment before the normalization of similarity scores, we check the average similarity score with,

$$\bar{s}_x = \frac{1}{k}\sum_{i=1}^{k} s(G_{M_i}, G_C|x) \tag{3.3.6}$$

We continue with the ensemble of predictions only if $\bar{s}_x$ is larger than a threshold δ, and we will discuss the choice of δ in the evaluation chapter.

Putting it all together: Algorithm 1 summarizes our transfer learning algorithm for fully automated sensor type mapping across buildings. We start from training a few base classifiers based on the data features and labeled instances in a source building. Then we generate clusters using GMM with DP priors, as detailed in Section 3.2 on the name features of instances in the target building. For each instance x in the target building, we measure the local similarity score for each base classifier. If the average similarity is significant enough, we compute the weight for each classifier at x by normalizing the similarity score. Finally we calculate the weighted sum of predictions from all base classifiers and obtain the label y for x.

Algorithm 1: Transfer Learning for Automated Mapping

```
Input: Data features of the source building $\mathcal{D}_\mathcal{S} = \{x_1^D, x_2^D, \dots, x_n^D\}$ and their labels
$\mathcal{Y}_\mathcal{S} = \{y_1, y_2, \dots, y_n\}$ data features of the target building $\mathcal{D}_\mathcal{T} = \{x_1^D, x_2^D, \dots, x_m^D\}$, and name
features of the target building $\mathcal{P}_\mathcal{T} = \{x_1^P, x_2^P, \dots, x_m^P\}$
Output: predicted labels $\mathcal{Y}$ of the instances in the target building
Initialize: Generate clusters with $DP(G_0, \eta)$ on name features $\mathcal{P}_\mathcal{T}$
Train $k$ classifiers $M_1, \dots, M_k$ based on $\mathcal{D}_\mathcal{S}$ and $\mathcal{Y}_\mathcal{S}$;
for $x^D$ in $\mathcal{D}_\mathcal{T}$ do
    Construct neighborhood graphs $G_M$ and $G_C$ for $x^D$ as defined in Section 3.3.2 for each $M_i$;
    Compute the similarity score for each $M_i$ with Eq. 3.3.5;
    Check the average similarity score $\bar{s}_x$ over all $M_i$ with Eq. 3.3.6;
    If $\bar{s}_x > \delta$, then use Eq. 3.3.4 and Eq. 3.3.2 to predict the label $y$;
end
```

3.4 Evaluation

To demonstrate the effectiveness of our proposed solution for automated sensor type mapping, we evaluate our transfer learning-based algorithms on data readings and point names of sensors collected from three distinct buildings. Extensive experimental comparisons confirm that the proposed fully automated technique is able to accurately classify sensor types for a considerable fraction of the instances without human intervention; with more training data, the performance of our technique can be further enhanced.

3.4.1 Background & Building Sensor Taxonomy

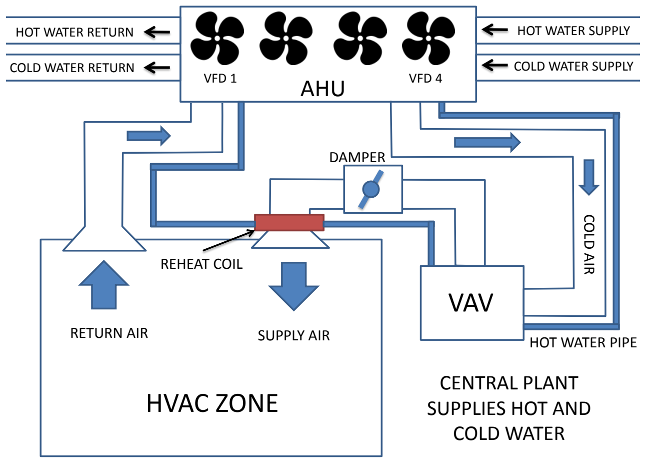


Figure 3.4: A typical HVAC system consisting of an air handler unit (AHU), several variable air volume boxes (VAV), water-based heating/cooling pipes and air circulation ducts. (Figure used with permission from the authors of [1].)

Figure 3.4 illustrates a typical heating, ventilation, and air conditioning (HVAC) system deployed in modern commercial buildings. An HVAC system usually uses a combination of hot and cold water pipes in conjunction with air handling units (AHU) to maintain the appropriate thermal environment within the building. An HVAC system usually consists of several AHUs and each AHU is responsible for a physical zone in the building. An AHU consists of variable speed drives that supply cold air (cooled by the supplied cold water) using ducts to VAV boxes distributed throughout the building. The hot water loop is also connected to these Variable Air Volume (VAV) boxes using separate pipes. Each VAV box controls the amount of air to be let into an HVAC zone using dampers, whose opening angle can be programmed. A reheat coil, which uses supplied hot water, is used to heat the air to meet the appropriate HVAC settings for each zone.

Table 3.2 summarizes all the types of sensors evaluated in these three buildings and the number of sensors of each type. For example, "room temperature" measures the temperature in room and for a better understanding, all the other temperature measurements on water circulation and air ventilation are illustrated in Figure 3.4. For setpoints, we assign only one general type which includes all set points for every actuator configured in the building.

Our evaluation dataset, which contains both the data readings and point names of sensor streams, is collected from over 2,500 sensors of more than 20 different types deployed in three commercial buildings. We collected a week's sensor reading data from each building. Specifically, building A is Rice Hall at the University of Virginia, where the sensing points report to the database in a Trane BMS, every 10 seconds to 10 minutes. Both building B and C are from UC Berkeley: building B is Sutardja Dai Hall, which contains sensors and equipment from KETI[2] and Siemens. Building C is Soda Hall, which uses an archaic system by Barrington Controls. Points and sensors in these two buildings transmit data to an sMAP [96] archiver periodically between every 5 seconds to 10 minutes.

All of our learning and classification procedures are implemented with the scikit-learn [97] library, an open-source machine learning package implemented in Python.

3.4.2 Feature Transferability

We first examine how well the two different types of features explained in Section 3.1 perform in classifying sensor types when being applied across buildings, i.e., learning a classifier based on the features from one building and testing it on another. We expect the data features to be more generalizable than the name features since building environments are conditioned similarly while the sensor naming conventions can be vastly different. For example, temperature readings in a room will usually fall between 60-70 degrees, no matter in which building. In contrast, point name features might not transfer well due to various naming conventions as shown in Table 2.1.

[2] `http://www.keti.re.kr/`

	Building		
Type	A	B	C
CO_2	16	52	0
Humidity	54	52	0
Air Pressure	142	216	215
Room Temp	159	231	208
Facility Operation Status	59	72	41
Facility Control	0	138	403
Setpoint	140	486	229
Air Flow Volume	14	172	9
Damper Position	0	290	10
Fan Speed	0	25	15
HW Supply Temp	27	1	0
HW Return Temp	15	1	0
CW Supply Temp	18	2	11
CW Return Temp	15	3	10
Supply Air Temp	20	17	3
Return Air Temp	6	2	4
Mixed Air Temp	5	2	3
Ice Tank Entering Temp	1	2	0
Ice Tank Leaving Temp	1	4	0
Occupancy	25	52	0
Timer	0	0	15
Sum	575	1124	1166

Table 3.2: Number of points by type for the 3 test buildings. “Temp” stands for “temperature”, “HW” for “hot water” and “CW” for “cold water”.

To examine how well each type of features can be used for knowledge transfer, we perform sensor type classification across buildings with each type of features separately. For example, with data features from building A, we train a random forest and apply it to building B on the same type of feature, and vice versa. Table 3.3 summarizes the results, from which we conclude that the data features are more useful to build a classifier across buildings than the name features, though the resulting performance is not perfect. We also conducted this experiment with other statistical classifiers, but the choice of classifiers does not affect the conclusion. These observations confirm our assumption that data feature is preferable to train statistic classifiers across buildings.

3.4.3 Features for Clustering

As shown in Algorithm 2, to perform classification in a target building we need to generate clusters for the points in it. These clusters are employed to compute the weight of base classifiers. Therefore, it is preferred to have points in the same cluster well align with their true class labels: the higher quality of clusters is, the more accurate the weights will be given to the base classifiers.

	Data Feature	Name Feature
A → B	0.778	0.400
B → A	0.612	0.254
B → C	0.521	0.030
C → B	0.399	0.302
A → C	0.922	0.021
C → A	0.584	0.399

Table 3.3: Type classification accuracy between two buildings (denoted as X → Y) with different sets of features: data features transfer better than point name features of sensors.

	Data Feature	Name Feature
A	0.21	0.58
B	0.34	0.75
C	0.32	0.78

Table 3.4: Quality of clusters generated on three buildings with different features measured by adjusted rand index (in the range of [0,1], higher is better).

Table 3.4 shows the quality of clusters generated by our non-parametric Bayesian method on the data features and the name features. Clustering quality is measured by adjusted rand index [98], which is a standard measure of the similarity between the grouping in clusters and the true labels. From the results we can clearly observe that clusters generated using the name features are more consistent with their true sensor type labels than those generated by the data features. This confirms our assumption about using the name features to estimate local cluster structures in target buildings.

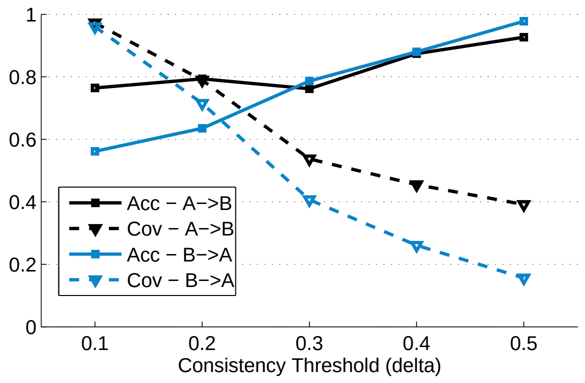
	Target A	B	C
Source A	N/A	0.754/0.496/0.510	0.921/0.766/0.538
B	0.614/0.228/0.362	N/A	0.513/0.247/0.258
C	0.582/0.299/0.421	0.393/0.158/0.190	N/A

Table 3.5: Base classifier performance across buildings on the data features. In each cell, the three numbers are the accuracy for random forest, logistic regression and SVM, respectively.

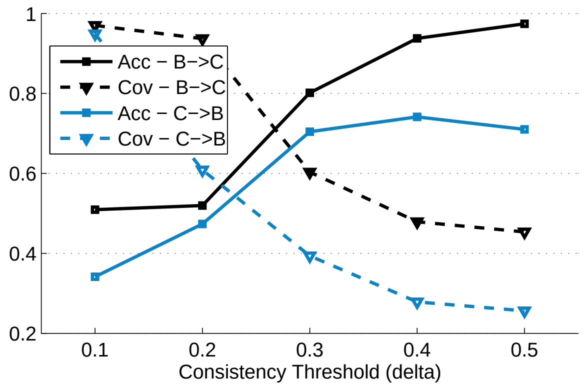
3.4.4 Base Classifier Performance

Our transfer learning based approach employs a few base classifiers. Each base classifier is trained on the same set of data features from the source building. There is no restriction on what classifiers should be used in this step. We employ three base classifiers: random forest, logistic regression (LR) and support vector machines (SVM) with RBF kernels.

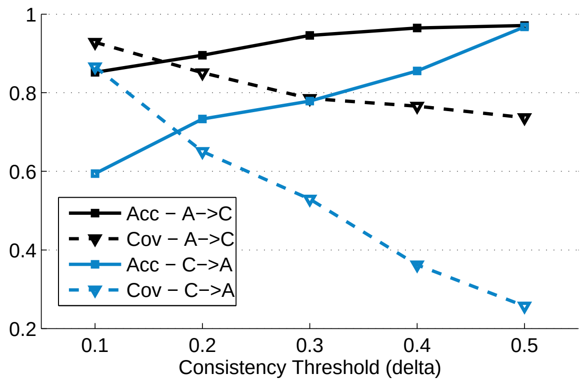
For the three buildings, we create six cross building training and testing pairs, and the corresponding classification performance is shown in Table 3.5. The base classifiers achieve an accuracy between 0.158 and 0.921 in this cross building evaluation scenario. On average, random forest performed the best, followed



(a) A and B



(b) B and C



(c) A and C

Figure 3.5: *Building Adapter*'s type classification accuracy (Acc) against labeled percentage (Cov) with transfer learning between different pairs of buildings (denoted as X $\rightarrow$ Y). As we increase the threshold, the coverage drops while the overall accuracy increases.

by SVM and LR. We should note that the accuracy of learning on building C and testing on building B is substantially worse than the other cases. One reason is that building B contains many dynamic temperature setpoints whose values change throughout the day, whereas the setpoints in building C are static. Anther major source of error is the type of air flow, whose reading amplitude is significantly different from the ones in building C.

3.4.5 Transfer Learning Performance

We first consider the case where only one building is exploited as the transfer source, i.e., all the base classifiers are trained on data from only one building. The overall accuracy of transfer learning for type classification across our three target buildings is illustrated in Figure 3.5. There is an intrinsic trade-off between the prediction accuracy and the percentage being labeled, as we set a threshold on the average weight of base classifiers. Since the average similarity score $\bar{s}$ reflects the confidence in predictions by classifiers, when we increase the threshold δ on the average similarity score for base classifiers, we filter out more instances with low prediction confidence and have labels of better quality – naturally resulting in a drop in the percentage of instances being labeled. We observe such a trend from Figure 3.5: when we have a small threshold, the approach can label more instances with low confident labels while we can label fewer instances with much better quality when increasing the threshold. Empirically, we can set the threshold δ be around 0.4, which strikes a reasonable balance between accuracy and coverage.

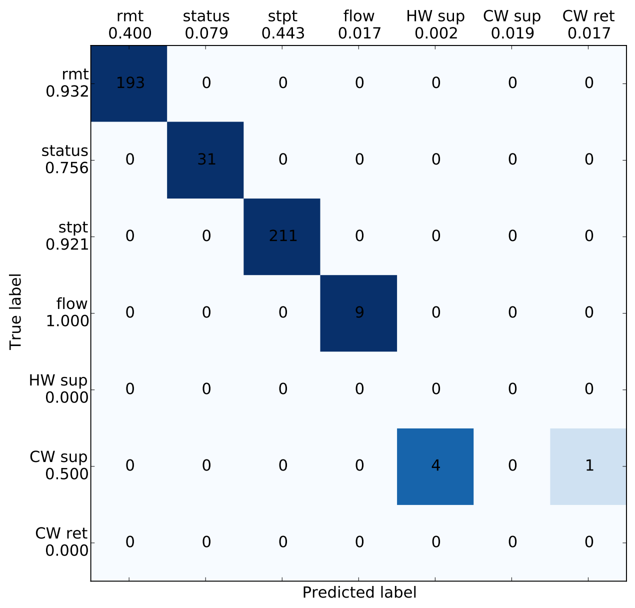


Figure 3.6: Confusion matrix for the transfer learning classification results for transferring from A to C. “rmt” stands for room temperature, “stpt” for setpoint, “sup” for supply, “ret” for return, and “SAT” for supply air temperature.

As an example, Figure 3.6 presents a confusion matrix for the classification results for the case of transferring from building A to building C. The numbers below the predicted labels on the x-axis denote the percentage of each type of sensor streams in that building, while the numbers below the true labels on the left denote the percentage of points being labeled by our technique for each sensor type.

On imbalanced data sets, investigation of accuracy alone is not enough. We also measure the weighted macro F1 score of classification for our approach and the baseline. The weighted macro F1 score is an altered version of macro F1 score [99], which calculates the F1 score for each class; where “one- versus-all” binary classification is performed and weighs the resulting F1 of each class by support (the number of true instances for each label).

	Target A		B		C	
	Acc	F1	Acc	F1	Acc	F1
Source A	N/A	N/A	0.943/0.934	0.936/0.931	0.977/0.970	0.981/0.971
B	0.897/0.875	0.932/0.913	N/A	N/A	0.950/0.952	0.939/0.937
C	0.862/0.862	0.864/0.864	0.726/0.702	0.691/0.726	N/A	N/A

Table 3.6: Accuracy and F1 score of transfer learning and the best base classifier with δ=0.4. Each cell contains two numbers for our approach and the baseline respectively. Overall, our approach outperforms the baseline.

As a baseline, we take the subset of instances in the new building that get labeled by the transfer learning process and apply base classifiers to predict labels on the same population. We set δ to 0.4 and repeat 10 times for the experiment in each direction (e.g., A$\rightarrow$ B). The average performance is reported in Table 3.6. Each cell contains two results: the former is from our approach while the latter is from the best base classifier. The case of transferring from C to B is again substantially worse than the other cases, this is because the transfer learning algorithm is bounded by the performance of base classifiers. Intuitively, if none of the base classifiers is able to correctly predict an instance, no matter how we manipulate the weights, it will not make a difference in the results. Overall, our approach outperforms the best baseline.

3.4.6 Training on Multiple Buildings

The above results indicate that our fully automated technique can label a fraction of the points in a commercial building with no human intervention. We further investigate how using multiple buildings as the training source would affect the labeling performance.

In practice, it is common to have labels for multiple buildings and combining them as a single training source might achieve better performance. We set δ to 0.4 and use two buildings as the training source. We repeat 10 times for the experiment on each target building and the average of 10 runs is reported. From Table 3.7, we see that combining multiple buildings as the source is superior to single source.

	Two Sources		Single Source	
	Acc	Cov	Acc	Cov
Target A	**0.863***	**0.397***	0.855	0.362
B	**0.899***	**0.458**	0.874	0.455
C	**0.980***	**0.890***	0.968	0.809

$^*p$-value<0.01

Table 3.7: Combining the knowledge from multiple buildings to infer on a new one is superior to exploiting one source building in both accuracy (Acc) and coverage (Cov).

Upon further inspection of the model predictions, compared with the case of single source, we observe that the improvements for integrating multiple sources in transfer learning mostly come from labeling more instances of the same classes, rather than capturing more new classes. Training based on the labeled data from multiple buildings produces more robust classifiers and as hundreds of buildings are mapped, our approach could become even more promising. In contrast, existing approaches scale only linearly with the number of buildings being mapped since they all require manual labeling and apply to only a single building.

3.5 Summary

In this chapter, we describe the design and evaluation of *Building Adapter* — a technique to automatically infer the type information in a building without manual labeling. This technique combines the complementary strengths of the data and the point name of a sensor to automatically create metadata for one building based on the information of another mapped building. By experimenting with over 2,500 streams from three buildings on two campuses, we demonstrate that our technique is able to automatically label more than 36% percent of the points in a new building with at least 85% accuracy, and in some case up to 81% points with more than 96% accuracy. We also illustrate how combining multiple buildings as the training source could further boost the performance, which shows promise of the technique. As a first step towards automatic metadata mapping, though not being able to label all the points, Building Adapter can label most common types of sensors related to HVAC with high accuracy. We will discuss in the next chapter a technique to complement Building Adapter, which exploits a minimal amount of human input to predict the type for sensors.

Chapter 4

Semi-Automated Type Classification

The fully automated technique discussed in the previous chapter does not create type labels for all the points. We therefore need to address the unlabeled points requiring some extent of human input. However, acquiring human labels is often costly, so we want to minimize the amount of labels we need from a human. Therefore, we develop a novel clustering-based active learning algorithm to perform semi-automated sensor type classification requiring only a *minimal* amount of human labels, within a single commercial building. This chapter describes in detail the design and evaluation of the algorithm.

4.1 Active Learning-based Classification

The idea underlying active learning is that a learning algorithm can achieve better performance with fewer training examples if it can choose the examples it learns from. We should note that although the naming schemata vary significantly across buildings, there will be definite variants within the same building, given there are only finite types of sensors deployed in a building. As a result, beyond the traditional active learning solutions [34, 35, 100], we further accelerate the learnin process by exploiting the clustering structure of *point names*. In particular, during the active learning procedure, unlabeled examples can be clustered; those in the same cluster are more likely to share the same label and hence do not need to be queried repeatedly. The cluster structure is therefore exploited to assist the manual labeling of instances and to avoid repeated selection of similar instances. Moreover, the acquired labels can be further propagated to the unlabeled neighbors in the same cluster to expedite classifier training, which is uniquely explored in our method, compared to previous active learning-style solutions [136, 16]. In our solution, the examples selected for labeling are chosen based on both their representativeness in the cluster and the informativeness of the cluster itself. A Gaussian Mixture Model with Dirichlet Process Prior [101] is used to cluster the instances on the fly to accommodate the dynamic nature of active learning. Label propagation is performed with respect to the connectivity of labeled examples' neighborhood in an adaptive manner. Because in practice one can get direct access to all point names in a target building, our active learning-based approach adopts the *pool-based* sampling setting, which selects the most informative instances from the entire collection of unlabeled data points. We shall note that both of the two approaches explained in this chapter and the previous chapter exploit the same set of attributes of sensors, yet in very different manners: the first approach looks for a "match" between the structure found in the name features and the data features, while the second approach only uses the structure in the name features to propagate manual labels to similar neighbors.

4.1.1 Clustering-based Active Learning

Formally, the problem of using active learning for semi-automated sensor type classification can be described as follows. For a given collection of unlabeled point represented with name features $P = \{x_1^P, x_2^P, \ldots, x_n^P\}$, in which the text string based point name x_i is represented as a d dimensional feature vector x_i^P (as explained in Section 3.1.2), we aim at learning a classifier $f : f(x^P) \rightarrow y$, with respect to a set of labeled instances $D_l = \{(x_1^P, y_1), (x_2^P, y_2), \ldots, (x_n^P, y_n)\}$ acquired during classifier training. In particular, y_i is the true sensor type for the point name x_i and takes value from a set of predefined labels. Our goal of learning is to maximize f's classification accuracy on the future testing data while minimizing the size of D_l. We should note that our proposed solution has no assumption about the classifier f; any supervised multi-class classifier can be used in our method.

Conventional active learning methods mostly focus on efficient search through the hypothesis space. Each time a new label is added to D_l, the set of hypothesis space shrinks, e.g., filters the classifiers that are inconsistent with the labels seen so far. Great research attention has been devoted to designing effective query strategies for label acquisition. Typical solutions include uncertainty-based sampling [35], query by committee [36], and expected model change [37].

However, such solutions implicitly assume unlabeled instances are independent, and thus fail to exploit any information conveyed in the density of data, i.e., the marginal distribution of $p(x^P)$ is assumed to be uniform. However, if the unlabeled instances in P form clusters, i.e., their clustering membership is consistent with the underlying class labels, one will only need one label from each cluster to estimate a perfect classifier. Although this example is overly optimistic, the clustering structure can still unveil the representativeness of instances with respect to their neighborhood. Hence, special emphasis should be given to those most representative instances for manual labeling. In addition, because instances in the same cluster are more likely to share the same label [102], one can accelerate active learning by avoiding labeling instances from the same cluster and propagating manual labels to nearby unlabeled instances.

In our solution, the density of unlabeled instances, i.e., $p(x^P)$, is exploited via its clustering structure. In particular, again, we use a probabilistic mixture model to identify the latent clusters, e.g., $p(x^P) = \sum_c p(x^P|c)p(c)$, where c denotes a cluster label. The detailed instantiation of this mixture model, e.g., how to decide the cluster size and the specification of cluster conditional likelihood, has been discussed in Section 3.2. With the identified clustering structure at hand, we devise a divide-and-conquer strategy for selecting the query instances: 1) it should come from the most *informative* cluster; and 2) it should be *representative* in that cluster. This query strategy focuses on the entire input space; comparing to those individual instance based selection methods, it can thus help to avoid querying outliers or repeatedly querying similar instances.

To quantify the "informativeness" of a cluster, we exploit an information theoretic metric - class entropy $H(c)$. For every cluster c, we apply $f(x^P)$ to predict the labels for all the unlabeled instances in it. Then based on these predicted labels, we compute the class entropy for this cluster as,

$$H(c) = -\sum_{y \in Y_c} p(y) \log p(y) \tag{4.1.1}$$

where Y_{c_i} is the set of unique labels in cluster c_i predicted by classifier $f(x^P)$.

A cluster with larger class entropy indicates increased discrepancy between the current classifier's prediction and clustering structure inferred from the density of unlabeled instances. Therefore instances from such a cluster are considered potentially more helpful in introducing new information to classifier training. In addition, the size of a cluster is also an important criterion for measuring its informativeness. When multiple clusters have similar class entropy, a larger cluster will indicate more uncertainty of the class labels in it. Hence acquiring a label for such a cluster can mostly reduce the classifier's uncertainty on a larger portion of instances. Combining these two aspects, we locate the cluster of choice by the product of cluster proportion $p(c)$ and class entropy $H(c)$ as follows,

$$\hat{c} = \arg\max_c p(c)H(c) \tag{4.1.2}$$

Once we locate the candidate cluster $\hat{c}$, we need to choose the most "representative" instance from it for labeling. We use the conditional likelihood $p(x^P|\hat{c})$ over all the unlabeled instances to select such an instance. The intuition behind this choice is that the instance with the maximum conditional likelihood best captures the homogeneity of instances in the selected cluster, such that knowing its label will provide a substantial boost for the classifier to predict the class labels within this cluster.

In addition, we view high class entropy in the selected cluster also as an indicator of low resolution of clustering results in that local region: classification boundary goes inside the cluster. To reduce this divergence between predicted classes and clustering results, we need to further separate this cluster into finer clusters. The benefit of this sub-clustering is that the class distribution estimated by the classifier is introduced into clustering; this helps generate more homogeneous clusters for later instance selection.

4.1.2 Label Propagation

The basic assumption in our clustering-based active learning solution is that instances in the same cluster tend to share the same class label. The query strategy described in the previous section exploits this assumption to avoid repeated selection of similar instances. In this section, we will further capitalize on this assumption to propagate the acquired labels to their nearby unlabeled neighbors to reinforce our current knowledge about the underlying class distribution.

The idea of label propagation is popularized in transductive learning [102, 103, 104], where class labels are propagated from the labeled instances to their unlabeled neighbors based on the structure of data manifold. In practice, the data manifold is approximated by the nearest neighbors of each instance derived from data features. To empirically validate the feasibility of applying label propagation in our problem, we randomly selected a small portion of labeled sensor point names in our evaluation corpus, computed the Euclidean distance between all pairs of instances based on their feature vectors, and grouped them into pairs from the same class and different classes. We plot the cumulative distributions of the three types of distances in Figure 4.1.

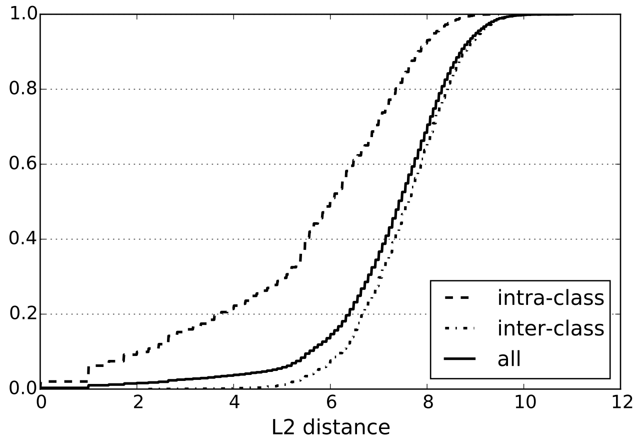


Figure 4.1: Distribution of pairwise distance between instances from the same class and different classes.

As we can clearly notice from the results, there is a clear gap between the cumulative distribution of distances between instances from the same class and those from different classes. This result also confirms our assumption in the proposed clustering-based active learning that nearby instances tend to share the same class label. Therefore, with properly constructed data manifold, we can confidently propagate the labels to their closest unlabeled neighbors and use them to update the classifier. This label propagation will amplify the importance of acquired labels and better avoid repeatedly querying of similar instances.

A typical approach to constructing the data manifold for label propagation in transductive learning is to look for the k nearest neighbors of each instance. However, in our solution it is challenging to find a universal setting of k. To accommodate the clustering structure, label propagation should only be performed within the same cluster. However, since the resulting clusters vary and change across iterations, a fixed setting of k might lead to inconsistency between the data manifold and clustering results. To address this issue, we introduce a distance threshold r to define the data manifold for label propagation in our active learning process. When the label y is acquired for instance x, all the unlabeled instances located within the distance r to x will be assigned the same label y. Those instances will be then removed from P and added to a collection of propagated label set D_p for later classifier training.

The optimal threshold r should be the minimum inter-class distance between any pair of instances. However, it is impossible to calculate without knowing the true class labels. In our solution, we keep an estimation of r from all the labeled instances in D_l during active learning,

$$r = \underset{(x_i^P, y_i),(x_j^P, y_j)\in D_l}{\arg\min} \frac{d(x_i^P, x_j^P)}{2}, \ with \ \ y_i \neq y_j \tag{4.1.3}$$

$d(x_i^P, x_j^P)$ is the Euclidean distance between two name feature vectors x_i^P and x_j^P.

We divide the minimum distance by 2 to avoid possible overlapping of label propagation between two labeled instances. The adaptive estimation of distance threshold defined in Eq (4.1.3) will shrink during active learning and therefore refines the propagated labels. Accordingly, we need to correct previously propagated labels where the estimation of r was less accurate. In particular, when we have a new estimation of r, we will remove instances from D_p that fall outside the region defined by the latest r to any closest labeled neighbors with the same label, and put these instances back to the unlabeled set P. This correction happens before the new round of classifier training.

Putting it all together: Algorithm 2 summarizes our clustering-based active learning algorithm for the semi-automated mapping. In each iteration, we select the most "informative" cluster measured by the class entropy of predicted labels by the classifier. We acquire a label for the instance centered in the selected cluster, and update the estimated distance threshold r for label propagation. We then update all previously propagated labels and propagate the newly obtained label to its unlabeled neighbors within the new estimated distance. At the end of this iteration, we perform sub-clustering on the selected cluster to refine its local clustering structure for later instance selection.

Algorithm 2: Clustering-based Active Learning for Semi-Automated Mapping

```
Input: point names $\mathcal{P} = \{x_1^P, x_2^P, \ldots, x_n^P\}$, and label budget $B$
Output: predicted labels of the point names $Y$
Initialize: Generate clusters with $DP(G_0, \eta)$ on $\mathcal{P}$, reset the labeled instance set $\mathcal{D}_l$ and propagated
  label set $\mathcal{D}_p$ to empty
while $B > 0$ do
    Train the classifier $f(x^P) \rightarrow y$ based on $\mathcal{D}_l$ and $\mathcal{D}_p$;
    Apply $f(x^P)$ to all instances in $\mathcal{P}$;
    Compute class entropy $H(c)$ defined in Eq (4.1.2) for each cluster $c$;
    Retrieve a cluster by $\hat{c} = \arg\max_c p(c)H(c)$;
    Acquire label $\hat{y}$ for example $\hat{x}$ given by $\arg\max_{x \in \mathcal{P}} p(x^P|\hat{c})$, and move $(\hat{x}, \hat{y})$ from $\mathcal{P}$ to $\mathcal{D}_l$;
    Update the distance threshold $r$ by Eq (4.1.3);
    Update all previous propagated label in $\mathcal{D}_p$ with new threshold $r$;
    Assign $\hat{y}$ to all unlabeled examples $x_u$, where $x_u \in \hat{c}$ and $d(x_u, \hat{x}) < r$;
    Move $(x_u, y)$ from $\mathcal{P}$ to $\mathcal{D}_p$;
    Perform sub-clustering with $DP(G_0, \eta)$ in $\hat{c}$;
end
```

4.1.3 Discussion

Comparing to the existing solutions for sensor type classification, our proposed algorithm addresses the problem from a totally different perspective. As we discussed before, because existing solutions [12, 136] isolated the annotation process from model training, it is likely to lead to wasted effort in data annotation. With active learning, we can restrict our effort to a smaller set of instances that are the most helpful for classifier training. A practical benefit of this learning approach is that we can rapidly bootstrap the building analytic software across different sites. Instead of going back and forth for several rounds of blind annotation, a building manager only needs to interact with the system for several iterations on the fly to achieve immediate satisfactory results.

Comparing to other active learning algorithms, our proposed method focuses on exploiting information conveyed in the data clustering structure. Based on the assumption that nearby instances are more likely to share the same class label, we select the most informative instances for labeling based on the identified latent clustering structure of instances and propagate the labels to their adjacent unlabeled neighbors. This helps us avoid repeatedly querying similar instances and enhance the importance of labeled instances. In addition, the clustering structure and label propagation strategy are adapted in an online fashion. In Nguyen and Smeulders's work [38], clustering is also exploited for active learning; but their clustering structure is static and no label propagation is performed. In Dasgupta and Hsu's work [39], hierarchical clustering is used to provide a more flexible and dynamic clustering structure, but no label propagation is performed either. We should note this proposed active learning algorithm is general, it only assumes the proximity of label distribution in nearby instances. Therefore, this algorithm can also be applied to a broader context, e.g., document categorization [105] and image retrieval [106].

4.2 Evaluation

In this section, we present the evaluation of the proposed algorithm using the same dataset as in Section 3.4.1. Via extensive experiments, we demonstrate that the active learning-based semi-automated classification method can achieve the same classification accuracy with much fewer labeled instances than all the baseline methods. To demonstrate the promise of the technique, we combine transfer learning with the active learning process to further accelerate the learning efficiency. We also demonstrate how metadata mapping can enable meaningful analytics on top of raw sensor streams under the context of commercial building energy control and comfort assessment.

4.2.1 Baselines

To evaluate the performance of our proposed active learning algorithm, we adopt four active learning algorithms as baselines.

Random (RAND): this method selects an example at random uniformly from the unlabeled set in each iteration.

Least Margin (LM) [107]: this method adopts the a simple yet effective sampling strategy, which queries the instances with least for labeling confidence measured by the difference of posterior probability of the first and second most probable class labels predicted by the classifier:

$$x_M^* = \arg\min_x \; p(\hat{y}_1|x) - p(\hat{y}_2|x),$$

where $\hat{y}_1$ and $\hat{y}_2$ are the first and second most probable class labels, respectively. We compute the class prediction probability by SVM based on the method proposed in [108].

Pre-clustering (PC) [38]: this method pre-clusters the instances and selects the example for querying satisfying two criteria: 1) locate at the classification boundary and 2) representative of dense clusters. The clusters are constructed by the same Gaussian Mixture Model with Dirichlet Process prior as used in our method.

Hierarchical Clustering (HC) [39]: this method leverages a hierarchical clustering structure, which is represented as a binary tree, and estimates the purity of labels for each cluster node. The algorithm iteratively selects instances from a subtree whose size is significantly large or the impurity of labels is high.

For all the baselines and our method, a linear SVM model is used as the classifier.

We measure the overall classification accuracy of each active learning algorithm with different amount of manual labels, and examine how many instances are needed by each method to reach a required accuracy level. Besides classification accuracy, we also compute the weighted macro F1 score of each method given different labeling budgets (e.g., 5%, 10%, etc of the entire unlabeled set) in each building.

In our experiments, to reduce possible bias introduced by training/testing split, we perform 10-fold cross validation for each active learning method, and repeat it 10 times with different random seeds. The average performance of 10 runs from each method is reported.

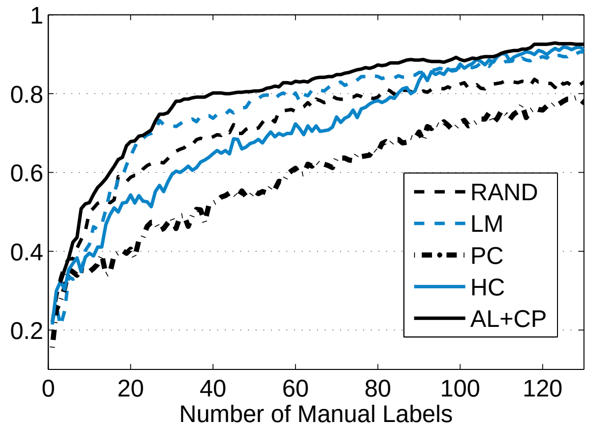
4.2.2 Classification Results

	Building A			Building B			Building C		
Labeled Percentage	1%	5%	10%	1%	5%	10%	1%	5%	10%
RAND	0.204	0.582	0.696	0.711	0.933	0.970	0.795	0.915	0.937
LM	0.222	0.572	0.769	0.736	0.995	0.995	0.835	0.980	0.988
PC	0.232	0.515	0.581	0.752	0.958	0.968	0.762	0.862	0.987
HC	0.326	0.516	0.669	0.829	0.993	0.996	0.867	0.977	0.987
AL+CP	**0.416***	**0.765***	**0.819***	**0.919***	**1.000***	**1.000***	**0.890***	**1.000***	**1.000***

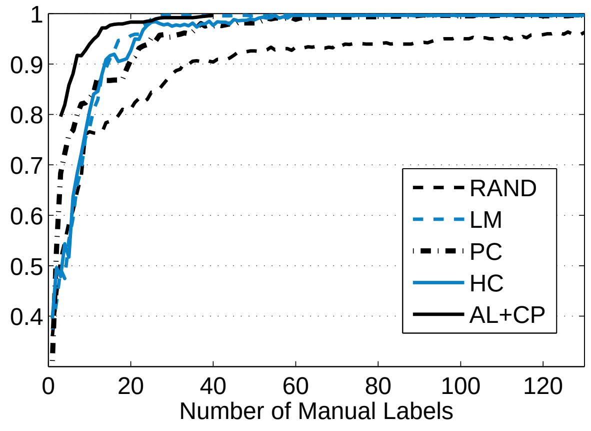
$^*p$-value<0.01

Table 4.1: Weighted macro F1 score of classification on three buildings with different labeling budgets. Our method (AL+CP) converges with less than 5% examples being labeled in Building B and C. It consistently outperforms the baselines in all the cases.

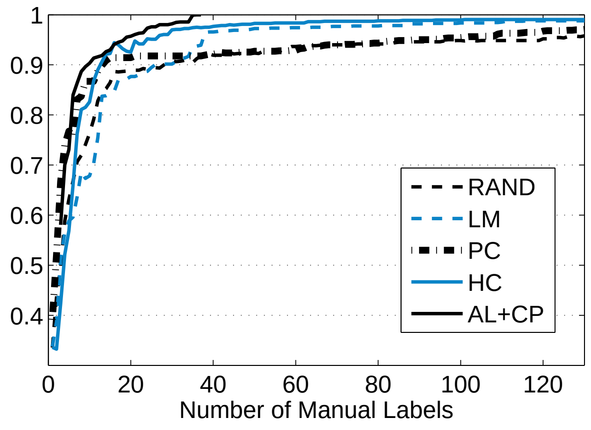
In Figure 4.2, we illustrate the comparison results of sensor type classification accuracy from all three buildings over all the methods. In all three buildings, our method performed the best against all the baselines. In particular, our method consistently requires the least amount of manual labels to achieve a satisfactory and converged accuracy. In building A, to achieve the accuracy of 90%, our clustering based active learning method (hereafter denoted as AL+CP) requires 105 labeled instances while HC needs 110, which is the best among all the baselines. LM takes 127 labeled instances and the other two baselines take a hundred more labeled instances to obtain the same accuracy. In building B, both our method and LM reach the accuracy of 99% with 26 instances, while it takes 51, 61 and more than 200 instances, respectively, for the HC, PC and RAND baselines. In building C, our method achieves 99% accuracy with 35 labels while HC and LM requires 101 and 135 labels, respectively, for the same accuracy. Again, PC and RAND require much more labeled instances to get the same performance.



(a) Building A



(b) Building B



(c) Building C

Figure 4.2: Classification accuracy on three different buildings: comparing to the baselines, our method (AL+CP) is able to achieve better accuracy on all buildings with less labeled examples.

On all three buildings, RAND performs reasonably on initial iterations (e.g., before 20) but becomes the worst later on. This is due to the imbalanced class distribution in the three buildings in our data set. RAND selects more instances from larger classes at the beginning and quickly gets a fraction of the major classes correctly classified. However, in later iterations, RAND has a lower chance of picking instances from smaller classes and converges to a local optimal quickly.

The PC baseline, which also exploits data clustering structure, performs significantly worse than other baselines for Building A. PC gives priority to the instances that are: 1) close to the decision boundary, and 2) representative of a cluster. Such criteria are very similar to those in our method. Further inspection reveals that instances selected by the PC baseline were neither informative nor representative enough to distinguish different types of instances in these buildings. This stems from the fact that for building A, the initial clusters from GMM fairly overlap with each other, which biases the PC method to select more instances with relatively high uncertainty but less representative in a cluster.

The HC baseline is quite effective in avoiding querying similar instances from dense areas. However, because HC does not perform label propagation, after all the clusters become roughly equally pure with sufficient sampling, it will start repeatedly querying previously sampled areas, which results in a long plateau for building B and C. For building A, the larger classes contain a few variations in their point names, therefore when HC avoids sampling from the same dense area at first, it misses important informative instances and is only able to pick them up in later iterations.

Moreover, we notice that, for all the methods, the convergence rate on building B and building C is much faster than building A, despite the fact that there are more points in these two buildings. The reason is that the point names in these two buildings share stronger regularities - the last segment always indicates the sensor type. Recall the examples shown in Table 2.1: `SDH_SF1_R282_RMT` from building B and `SODA1R410B_ART` from building C. Both of them adhere to the same order of segment categories - `building name-air handler/supply fan identifier-room number-sensor type`, only with a slightly different set of delimiters. Our string feature captures such regularity and helps the learning algorithm converge faster. However, the same rule does not apply to building A because the type information can be encoded in any segment in a point name. For instance, a temperature measurement can be named in many conventions, such as `averageSpaceTemperature 1st Floor Area1`, `East Space Temperature scc2UIP33` or `RM511A Zone Temp`. With such variety in the naming conventions, the learning algorithms need to explore all possible variants of point names for convergence in building A.

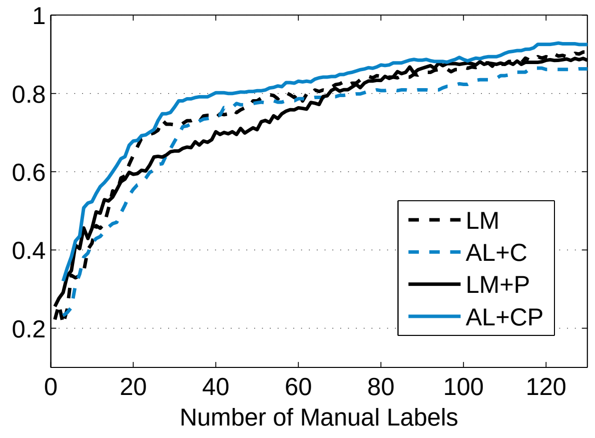
We also measure the weighted macro F1 score of classification for each method with different labeling budgets, i.e., different percentage of total instances that can be labeled. We examine the F1 score at three different labeling budgets, 1%, 5% and 10%. Paired two sample t-test is performed to validate the statistical significance of improvement from our method over the best-performing baseline. The results are shown in Table 4.1. In general, more labeled instances lead to better classification performance over all classes; and our method performed the best in all cases. As we discussed earlier, all methods converge much faster in building B and building C than in building A. In these two buildings, our method can achieve an 100% classification accuracy with slightly more than 1% of total instances being labeled.

In addition, to investigate the effect of data clustering and label propagation in our proposed active learning algorithm, we also conduct experiments by disabling data clustering and label propagation in our method. From the results in Figure 4.3, we can observe that with clustering only, the performance of our method is no better than margin-based active learning; and beyond a certain point, the performance plateaus because the classification boundary converges to the clustering structure, and no more new instances could be selected for labeling. However, we do observe some improvement from clustering only over margin-based active learning for building B and building C in early iterations. This is because the same type of points in these two buildings contain few variations and these initial instances at the center of relatively pure clusters are more representative for a dense region. Interestingly, after adding label propagation, we clearly see a boost in the performance than performing clustering only. We attribute the improvement of our method to label propagation which considerably amplifies the amount of available labeled instances for classifier training, and therefore helps to estimate a better classifier.

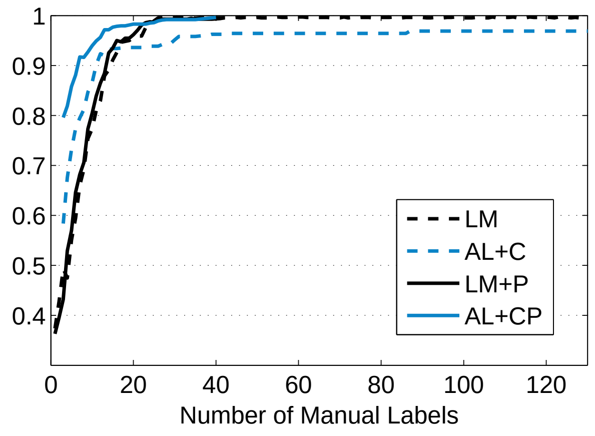
Besides, we also examine how the label propagation alone can help active learning in general. We adopt the same distance threshold estimation method as defined in Eq (4.1.3) to introduce propagated labels into the margin-based baseline and denote it as LM+P in Figure 4.3. On building A, the inclusion of label propagation makes LM even worse, because in early stage the estimated threshold r in LM+P is not accurate enough, and it mistakenly propagates the labels. Only when more labels are acquired in later stages, the estimation of distance threshold r get improved and therefore LM receives propagated labels with improved quality. In our method, because of data clustering, the labeled instances in the early stage are more representative; it in turns helps better estimate the distance threshold r. As a conclusion, data clustering and label propagation complement with each other in our proposed method, and help active learning acquire more informative instances quickly.

4.2.3 Combining the Fully and Semi-Automated Approaches

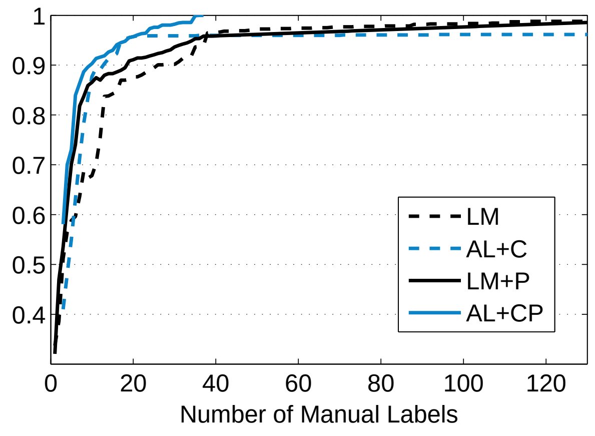
As we discussed earlier, the fully automated technique does not label all the points in a new building, so we can use the semi-automated approach to address the remaining unlabeled streams. To examine how well it works by joining the two approaches, we first apply our fully automated technique to automatically labels a fraction of the points in a building, and then we switch to the semi-automated algorithm to label the rest.



(a) Building A



(b) Building B



(c) Building C

Figure 4.3: Comparison of margin-based active learning (LM), clustering-based active learning without label propagation (AL+C), margin-based active learning with label propagation (LM+P), and clustering with label propagation based active learning (AL+CP). The label propagation significantly improves the performance of learning with clustering only.

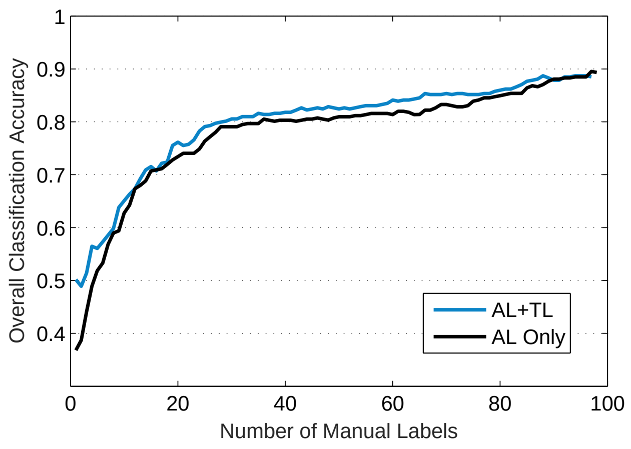


Figure 4.4: Our transfer learning based approach can complement traditional labeling technique to achieve better accuracy with the same amount of labels.

We adopt the active learning technique developed in Section 4.1 and as a comparison, we examine how much combining the two techniques can accelerate the semi-automated labeling process.

In the target building, we split all the testing instances into 10 folds. We run the active learning method (AL) and active learning combined with transfer learning method (AL+TL), respectively, on nine folds while testing on the one remaining fold. For the case of AL+TL, we start from the fully automated process with a δ = 0.6 to minimize the amount of inaccurate labels from transfer learning. Then we switch to the AL algorithm developed in Section 4.1 where we acquire one manual label per iteration. For brevity, we only show the results from transferring from building B to building A in Figure 4.4. We can see that AL+TL helps reduce the number of manual labels to achieve the same performance as AL. For example, given the same amount of manual labels, AL+TL can always outperform AL.

4.2.4 A Proof-of-Concept Application

As a first step towards automated metadata mapping, our algorithm is able to classify and transform the type information in the primitive metadata into a normalized name space. It provides a better opportunity for running uniform analysis on heterogeneous metadata across buildings with different management systems. As a proof-of-concept, we demonstrate an illustrative example, in which we search over the normalized metadata for different types of sensors that are in the same room under the application context of building comfort assessment and energy control.

Ideally, with the occupancy information of spaces considered, a properly configured building should automatically stop conditioning unoccupied rooms and areas. However, one typical problem plaguing buildings that incurs energy waste is unoccupied room being conditioned. For a building manager to inspect the building and locate such spots for better scheduling, he should be able to run simple keyword based searches over the metadata looking for different types of streams, such as room temperature and occupancy streams. With the desired types of streams returned, he can match different types of streams by room location to perform further examination, e.g., which room is still comfortable during unoccupied periods.

To accomplish the task, we first classify the sensor streams by type with both our method and the best baseline (LM). Then based on the predicted sensor type for each stream, we map each type of sensors into a common name space, e.g., all the streams predicted as room temperature are named as “room temperature”. With the mapped type information, we can simply retrieve all the streams of a certain type with one keyword, e.g, using “temperature” to search for temperature streams. The next step is to match different types of sensors by room location. In particular, we derive a few regular expressions to find the segment indicating room location in the point names of each sensor.

In this experiment, we search for four types of sensors - occupancy, temperature, humidity and CO_2 - by simply using these type names as the keywords. Then we match the room location of these four groups with regular expressions on their original primitive point names. Specifically, we examine the accuracy of three searches: 1) occupancy and temperature in the same room; 2) occupancy, temperature and humidity in the same room; and 3) occupancy, temperature, humidity and CO_2 in the same room, given these combinations are usually needed to assess the comfort of a room or identify potential waste in unoccupied rooms.

	AL+CP		LM	
	Precision	Recall	Precision	Recall
O	0.930	1.000	1.000	0.688
T	0.892	0.935	0.635	0.975
H	0.962	1.000	0.778	0.519
C	0.556	0.833	0.222	0.286

Table 4.2: The performance of searches over the normalized metadata with our method (AL+CP) and the best baseline (margin based active learning, LM). We search for four specific types of streams: occupancy (O), temperature (T), humidity (H) and CO_2 (C).

	AL+CP		LM	
	Precision	Recall	Precision	Recall
O+T	0.832	1.000	1.000	0.900
O+T+H	0.926	1.000	1.000	0.400
O+T+H+C	0.375	0.833	1.000	0.125

Table 4.3: The performance of searches for different types of streams that are in the same room. We consider the pair of occupancy and temperature (O+T), the combination of occupancy, temperature and humidity (O+T+H), and also the combination of occupancy, temperature, humidity along with CO_2 (O+T+H+C).

Table 4.2 illustrates the performance of basic searches for occupancy (O), temperature (T), humidity (H) and CO_2 (C). Our method can return better results with both higher recall and precision because of better predictions on type class for the streams. Given the four returned groups of streams, we further run regular expression based matching on the primitive metadata to find the pairs in the same room. For instance, we first identify the room location of each returned occupancy stream with regular expressions, then we search for the temperature stream with the same room number as each of these occupancy streams. Similarly, we further search for humidity and CO_2 streams in the same room. With these different types of streams grouped into the same room, a building manager can compare the actual data against some standard to decide if a room is problematic or not. We summarize the search results in Table 4.3. In general, normalized metadata by our method produces search results with higher recall and slightly lower precision. Such results are expected considering the practical demand from building managers - it is acceptable to return some unexpected results (false positives) while not missing the expected ones (false negatives), where he can manually examine a much smaller group of candidates and filter out the incorrect ones. We conclude that with normalized metadata for a building, people such as a building manager can more easily retrieve expected streams and conduct meaningful analysis to identify problems in a building.

4.3 Summary

In this chapter, we introduce a novel, effective yet general active learning method to address the sensor type inference problem, requiring the minimal amount of human input. This method can be used to complement the Building Adapter developed in the last chapter, or used separately as a standalone procedure. Following the assumption that similar instances are more likely to share the same class label, our solution exploits the data clustering structure and propagates the labels to their nearby unlabeled examples to accelerate the learning process. Extensive experimental comparisons are performed against several state-of-the-art active learning algorithms with over 20 different sensor types and 2,500 sensor streams collected from three buildings. The method is able to achieve more than 92% accuracy for type classification with at least 20% fewer labeled examples than baselines. The proposed method is general, and therefore it is applicable in a broader contexts such as document categorization [105] and image retrieval [106].

In our current metadata normalization process, only the text features from point names are utilized. However, another important aspect of the sensor streams is the actual time series data from sensors, i.e., the sensor readings. Features constructed from such raw signals can also be introduced to characterize the stream, and these feature will become vital when the point names are missing or corrupted. In addition, our current problem setting is limited to one building; it is necessary for us to solve this problem across buildings, e.g., classification model learned in one building can be used to bootstrap the learning in another building, and we discuss in more details about this in Chapter 7.

Chapter 5

Cross-Predictive Clustering for Sensor Type

We discussed an unsupervised transfer learning technique in Chapter 3 for inferring the sensor types, where we have implicitly assumed abundant data for the task. Specifically, we assumed the number of readings per sensor time series is more than the number of sensors. However, the proliferation of cheap, ubiquitous sensing infrastructure has pervaded and been monitoring the world, and many expect the Internet of Things to have over 25 billion devices by 2020 [109]. In this paradigm, sensory time series data will often be high-dimensional: the number of time series d (i.e., number of sensors) will be much larger than the length of each time series T. Conventional classification techniques are likely to fall short in such scenarios, and we tackle the problem from the perspective of time series clustering, which often serves as an important first step for many applications and poses long-standing challenges. In this chapter, we explore the challenge of time series clustering in the *high-dimensional* regime, in the context of recognizing sensor types.

The key to time series clustering is how to characterize the similarity between any two time series. In the past several decades, various metrics for measuring the similarity/distance between time series have been investigated [40, 41, 42, 43, 44, 45, 46, 47, 48, 49], and so on. Hidden Markov Models [50, 51] have also been utilized to derive the distances between time series for clustering. Recently, a few new metrics [52, 53] to measure the similarity between time series have been proposed and applied to cluster brain-computer interface data and motion capture data. However, all the aforementioned work either did not provide theoretical guarantees for their methods, or only considered scenarios where the number of observations per time series T far exceeds the number of time series d.

In this chapter, we explore a new similarity measure called "cross-predictability": the degree to which a future value in each time series is predicted by past values of the others. This measure captures causal relationships between time series, such as seasonal or diurnal effects on multiple environment sensors, market effects on multiple stock prices, and so on. However, it is challenging to estimate such cross-predictability among time series in the high-dimensional setting where $d > T$: a conventional regression task, for example, would have d variables and T equations, which is under-constrained. Intuitively, only time series in the same cluster would have significant cross-predictability for each other, thus yielding sparse relationships that are indicative of the cluster structure. Consequently, we propose to estimate cross-predictability by imposing a sparsity assumption on the cross-predictability matrix, i.e., that only time series in the same cluster have significant cross-predictability with each other. To do this, we propose a new regularized Dantzig selector, which is a variant of standard Dantzig selector [110], to estimate the similarity among the time series. We demonstrate that this approach is computationally attractive because it involves solving d regularized Dantzig selectors that can be optimized by alternating direction method of multipliers (ADMM) [111] in parallel.

Additionally, we provide a theoretical proof that the proposed algorithm will identify the correct clustering structure with high probability, if two conditions hold: 1) the individual time series themselves can be modeled with an autoregressive model [112], and 2) the transition matrix for the vector autoregressive model is block diagonal, i.e., that it is actually possible to create clusters such that time series in the same cluster are cross-predictive while those in different clusters are not.

To the best of our knowledge, this is the first practical high-dimensional time series clustering algorithm with a provable guarantee. It is worth noting that the proposed algorithm can be generally applied to cluster any high dimensional times series, regardless of the underlying data distribution. We make the autoregressive model assumption solely for the purpose of providing the theoretical guarantees for our method.

To demonstrate the effectiveness of our method, we conduct experiments on a real-world data set of sensor time series as well as simulations with synthetic data. Our method can achieve more than 80% clustering accuracy on the real-world data set, which is 20% higher than the state-of-art baselines.

Notations We compile here some standard notations used throughout this chapter. We use lowercase letters $x, y, \ldots$ to denote scalars, bold lowercase letters $\mathbf{x}, \mathbf{y}, \ldots$ for vectors, and bold uppercase letters $\mathbf{X}, \mathbf{Y}, \ldots$ for matrices. We denote random vectors by $\boldsymbol{X}, \boldsymbol{Y}$. We denote the (i, j) entry of a matrix as M_{ij}, and use $\mathbf{M}_{i*}$ to index the i-th row of a matrix (likewise, $\mathbf{M}_{*j}$ for the j-th column). We also use $\mathbf{M}_{S,T}$ to represent a submatrix of $\mathbf{M}$ with its rows indexed by the indices in set S and columns indexed by T. In addition, we write S^c to denote the complement of a set S. For any matrix $\mathbf{M}$, $\mathcal{P}(\mathbf{M})$ represents the symmetric convex hull of its columns, i.e., $\mathcal{P}(\mathbf{M}) = \text{conv}(\pm\mathbf{X})$. For any matrices $\mathbf{M}_1, \mathbf{M}_2, \ldots \mathbf{M}_k$, we denote a block diagonal matrix by $\text{diag}(\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_k)$ such that the k-th diagonal block is $\mathbf{M}_k$. Throughout the chapter, we will use vector norm ℓ_q for $0 < q < \infty$ and ℓ_∞ of $\mathbf{v}$ defined as $\|\mathbf{v}\|_q = \left(\sum_i |v_i|^q\right)^{1/q}$, $\|\mathbf{v}\|_{\infty,\infty} = \max_i |v_i|$, and matrix norm ℓ_q, element-wise ℓ_∞ and ℓ_F of $\mathbf{M}$ as $\|\mathbf{M}\|_q = \max_{\|\mathbf{v}\|_q=1} \|\mathbf{M}\mathbf{v}\|_q$, $\|\mathbf{M}\|_{\infty,\infty} = \max_{ij} |M_{ij}|$, $\|\mathbf{M}\|_F = \left(\sum_{i,j} |M_{ij}|^2\right)^{1/2}$.

5.1 The Cross-Predictive Clustering

5.1.1 The VAR Model

Our algorithm is motivated by the autoregressive model, and the later-on theoretical guarantee for our algorithm also relies on the autoregressive model assumption, so we briefly review the stationary first-order vector autoregressive model with Gaussian noise here. Let random vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T$ be from a stationary process $(\boldsymbol{X}_t)_{t=-\infty}^{\infty}$, and we further define $\mathbf{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t, \ldots \boldsymbol{X}_T]^\top \in \mathbb{R}^{T\times d}$, where $\boldsymbol{X}_t = (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$ is a d-dimensional vector and each column of $\mathbf{X}$ is a one-dimensional time series with T samples. In particular, we assume each $\boldsymbol{X}_t$ can be modeled by a first-order vector autoregressive model:

$$\boldsymbol{X}_{t+1} = \mathbf{A}\boldsymbol{X}_t + \boldsymbol{Z}_t, \text{ for } t = 1, 2, \ldots, T-1. \tag{5.1.1}$$

To secure the above process to be stationary, the transition matrix $\mathbf{A}$ must have bounded spectral norm, i.e., $\|\mathbf{A}\|_2 < 1$. We also assume $\boldsymbol{Z}_t \sim N(0, \boldsymbol{\Psi})$ is i.i.d. additive noise independent of $\boldsymbol{X}_t$, and $\boldsymbol{X}_t$ has zero mean and a covariance matrix $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{X}_t \sim N(0, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{X}_t\boldsymbol{X}_t^\top]$ is the autocovariance matrix. In addition, we have the lag-1 autocovariance matrix as $\boldsymbol{\Sigma}_1 = \mathbb{E}[\boldsymbol{X}_t\boldsymbol{X}_{t+1}^\top]$. Since $(\mathbf{X}_t)_{t=-\infty}^{\infty}$ is stationary, it is easy to observe that the covariance matrix $\boldsymbol{\Sigma}$ depends on $\mathbf{A}$ and $\boldsymbol{\Psi}$, i.e., $\boldsymbol{\Sigma} = \mathbf{A}^\top\boldsymbol{\Sigma}\mathbf{A} + \boldsymbol{\Psi}$, and we further have:

$$\boldsymbol{\Sigma}\mathbf{A}^\top = \boldsymbol{\Sigma}_1. \tag{5.1.2}$$

Essentially, the zero and nonzero entries in the transition matrix $\mathbf{A}$ directly reflect the Granger non-causalities and causalities with regard to the stochastic time series. In other words, a nonzero entry A_{ij} implies that the j-th time series is predictive for the i-th time series, with the magnitude $|A_{ij}|$ indicating how much the predictive power is. The new similarity measure in our clustering algorithm is built upon such cross-predictive relationship between time series. Now we set to introduce the clustering algorithm.

5.1.2 The Proposed Clustering Algorithm

Our algorithm first estimates the cross-predictability among the time series, and then identifies the clustering structure based on the estimated relationship. To introduce our proposed algorithm, we need the following notations: $\mathbf{X}_\mathcal{S} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{T-1}]^\top \in \mathbb{R}^{(T-1)\times d}$, $\mathbf{X}_\mathcal{T} = [\boldsymbol{X}_2, \ldots, \boldsymbol{X}_T]^\top \in \mathbb{R}^{(T-1)\times d}$, $\hat{\mathbf{\Sigma}} = \mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S}/(T-1)$, and $\hat{\mathbf{\Sigma}}_1 = \mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{T}/(T-1)$. Inspired by the relationship in Eq. (5.1.2), our main idea is to estimate $\mathbf{A}$ based on the relationship between $\mathbf{A}$ and the autocovariance and lag-1 autocovariance matrices. This motivates the following Dantzig selector type estimator [113],

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \|\mathbf{A}\|_1 \quad \text{subject to} \quad \|\hat{\mathbf{\Sigma}}\mathbf{A}^\top - \hat{\mathbf{\Sigma}}_1\|_{\infty,\infty} \le \mu, \tag{5.1.3}$$

where $\mu > 0$ is a tuning parameter. Since each row of $\mathbf{A}$ is independent, the above optimization problem can be decomposed into d independent sub-problems and solved individually as follows:

$$\hat{\boldsymbol{\beta}}_i = \arg\min_{\boldsymbol{\beta}_i} \|\boldsymbol{\beta}_i\|_1 \quad \text{subject to} \quad \|\hat{\mathbf{\Sigma}}\boldsymbol{\beta}_i - \hat{\boldsymbol{\gamma}}_i\|_{\infty,\infty} \le \mu, \tag{5.1.4}$$

where $\hat{\boldsymbol{\gamma}}_i = (\hat{\mathbf{\Sigma}}_1)_{*i} = \mathbf{X}_\mathcal{S}^\top(\mathbf{X}_\mathcal{T})_{*i}/(T-1)$, i.e., $\hat{\boldsymbol{\gamma}}_i$ is the i-th column of $\hat{\mathbf{\Sigma}}_1$, and $\hat{\mathbf{A}} = \left[\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_d\right]^\top \in \mathbb{R}^{d\times d}$ with each $\hat{\boldsymbol{\beta}}_i \in \mathbb{R}^d$. Therefore, the $\hat{\boldsymbol{\beta}}_i$ in (5.1.4) is an estimation of the i-th row of the transition matrix $\mathbf{A}$. Furthermore, for each $\mu > 0$, there always exists a $\lambda > 0$ such that (5.1.4) is equivalent to the following regularized Dantzig selector type estimator:

$$\hat{\boldsymbol{\beta}}_i = \arg\min_{\boldsymbol{\beta}_i} \ \lambda\|\hat{\mathbf{\Sigma}}\boldsymbol{\beta}_i - \hat{\boldsymbol{\gamma}}_i\|_{\infty,\infty} + \|\boldsymbol{\beta}_i\|_1, \tag{5.1.5}$$

where λ is a regularization parameter to determine the sparsity of the estimation. (5.1.4) can be solved by alternating direction method of multipliers (ADMM) [111]. All the d optimization problems can be solved in parallel, thus computationally efficient.

After solving the problem in (5.1.5), we construct an affinity matrix $\mathbf{W}$ based on $\hat{\mathbf{A}} = \left[\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_d\right]^\top$ by symmetrization, and compute the corresponding Laplacian to perform standard spectral clustering [114, 115] to recover the clusters in the input time series. The procedure is summarized in Algorithm 3.

5.1.3 Discussion

At first glance, the regularized Dantzig selector in (5.1.5) and Lasso appear similar. However, different from Lasso, the "input" of the regression problem in Eq (5.1.5) is the lag-0 covariance matrix, and the "response" is the lag-1 covariance matrix. Here the lag-one covariance matrix encodes and includes into consideration the first-order temporal information, which is missing in conventional similarity metrics such as correlation. Additionally, different from the Lasso-based estimation procedure [116], which penalizes the square loss, the regularized Dantzig selector estimator penalizes the $\ell_{\infty,\infty}$ loss.

Algorithm 3: Time Series Clustering Algorithm

```
Input: Time series $\mathbf{X} = [\boldsymbol{X}_1, \dots \boldsymbol{X}_T]^\top \in \mathbb{R}^{T \times d}$, $\mathbf{X}_{\mathcal{S}} = [\boldsymbol{X}_1, \dots, \boldsymbol{X}_{T-1}]^\top \in \mathbb{R}^{(T-1) \times d}$,
$\mathbf{X}_{\mathcal{T}} = [\boldsymbol{X}_2, \dots, \boldsymbol{X}_T]^\top \in \mathbb{R}^{(T-1) \times d}$, $\hat{\boldsymbol{\Sigma}} = \mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}}/(T-1)$, and $\hat{\gamma}_i = \mathbf{X}_{\mathcal{S}}^\top (\mathbf{X}_{\mathcal{T}})_{*i}/(T-1)$
Output: Cluster membership of each time series $\boldsymbol{Y}$
1. Solve for each $i = 1, \dots, d$:
       $\hat{\boldsymbol{\beta}}_i = \arg\min_{\boldsymbol{\beta}_i} \lambda \|\hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}_i - \hat{\boldsymbol{\gamma}}_i\|_{\infty,\infty} + \|\boldsymbol{\beta}_i\|_1;$
2. Set $\hat{\mathbf{A}} = \left[\hat{\boldsymbol{\beta}}_1, \dots, \hat{\boldsymbol{\beta}}_d\right]^\top$;
3. Construct the affinity graph $G$ with nodes being the $d$ time series in $\mathbf{X}$, and edge weights given by
   the matrix $\mathbf{W} = |\hat{\mathbf{A}}| + |\hat{\mathbf{A}}|^\top$;
4. Compute the unnormalized Laplacian $\mathbf{L} = \mathbf{M} - \mathbf{W}$ of graph $G$, with $\mathbf{M} = \mathrm{diag}(m_1, m_2, \dots, m_d)$
   and $m_i = \sum_{j=1}^d W_{ij}$;
5. Compute the first $k$ eigenvectors $\sigma_1, \dots, \sigma_k$ of $\mathbf{L}$ and let $\mathbf{V} \in \mathbb{R}^{d \times k}$ be the matrix containing as
   columns the first $k$ eigenvectors;
6. Cluster time series $\mathbf{x}'_i \in \mathbb{R}^k$, as the $i$-th row of $\mathbf{V}$, with the $k$-means algorithm into clusters
   $\mathcal{C}_1, \dots, \mathcal{C}_l, \dots, \mathcal{C}_k$.
```

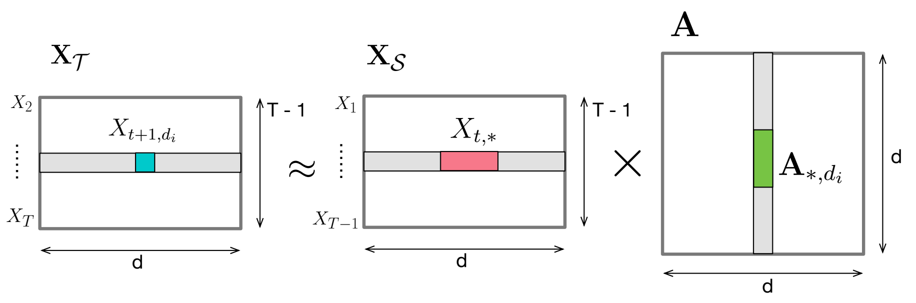


Figure 5.1: Illustration of the Proposed Regularized Dantzig Selector: it solves the regression problem to estimate the predictive relationship between the observations from time $t+1$ and time t considering all the time series.

It is also worth noting that Algorithm 3 shares a similar high level idea as the subspace clustering (SC) algorithm [117, 59, 58], but the key difference is that SC considers the relationship between each data point and all the other points, while in contrast, our estimator solves the regression problem to estimate the predictive relationship between the observations from time $t+1$ and the observations from time t considering all the time series, as illustrated in Figure 5.1. Another fundamental difference here is, SC assumes data are i.i.d. and lie on different subspaces, while here the time series data are obviously dependent. This poses a big challenge to the theoretical analysis of our algorithm.

5.2 Main Results

In this section, we state our main theory - a provable guarantee for successfully recovering the underlying clustering structure of the input time series. We first introduce some necessary definitions for understanding our main theorem.

5.2.1 Preliminaries

To define the clusters among time series $\mathbf{X}$ under the context of VAR model, we assume $\mathbf{A} = \operatorname{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_l, \ldots, \mathbf{A}_k)$ to be block diagonal, where $\mathbf{A}_l \in \mathbb{R}^{d_l \times d_l}$ and the number of time series d satisfies $d = \sum_{l=1}^{k} d_l$. Consequently, we can rewrite $\mathbf{X}$ as $\mathbf{X} = \left[\mathbf{X}^1, \ldots, \mathbf{X}^l, \ldots, \mathbf{X}^k\right]$ with each $\mathbf{X}^l \in \mathbb{R}^{T \times d_l}$ obeying:

$$\boldsymbol{X}_{t+1}^l = \mathbf{A}_l \boldsymbol{X}_t^l + \boldsymbol{Z}_t^l, \text{ for } t = 1, 2, \ldots, T-1,$$

which essentially defines the clustering structure in the time series, such that the data $\boldsymbol{X}_{t+1}^l \in \mathbb{R}^{d_l}$ at time point $t+1$ depends only on the data $\boldsymbol{X}_t^l$ from the previous time point t in the same block indexed by $\mathbf{A}_l$. In other words, as an effect of $\mathbf{A}_l$, data are more predictive for each other in the same block, rather than for those in the other blocks. The block diagonal transition matrix $\mathbf{A}$ gives rise to the fact that the time series in $\mathbf{X} \in \mathbb{R}^{T \times d}$ formulate k clusters $\mathcal{C}_1, \ldots, \mathcal{C}_l, \ldots, \mathcal{C}_k$ of $\mathbb{R}^{T \times d_l}$, and each $\mathcal{C}_l$ contains d_l one-dimensional time series of $\mathbb{R}^T$ denoted as $\mathbf{X}^l$. Without loss of generality, let $\mathbf{X} = \left[\mathbf{X}^1, \ldots, \mathbf{X}^l, \ldots, \mathbf{X}^k\right]$ be ordered. We further write $\mathcal{S}_l$ to denote the set of indices corresponding to the columns of $\mathbf{X}$ that belong to cluster $\mathcal{C}_l$.

Definition 5.2.1 (Cluster Recovery Property)**.** *The clusters $\{\mathcal{C}_l\}_{l=1}^k$ and the time series $\mathbf{X}$ from these clusters obey the cluster recovery property (CRP) with a parameter λ, if and only if it holds that for all i, the optimal solution $\hat{\boldsymbol{\beta}}_i$ to* (5.1.5) *satisfies: (1) $\hat{\boldsymbol{\beta}}_i$ is nonzero; (2) the indices of nonzero entries in $\hat{\boldsymbol{\beta}}_i$ correspond to only the columns of $\mathbf{X}$ that are in the same cluster as $\mathbf{X}_{*i}$.*

This property ensures that the output coefficient matrix $\hat{\mathbf{A}}$ and affinity matrix $\mathbf{W}$ will be exactly block diagonal, with each cluster represented in a disjoint block. Particularly, recall that we assume the transition matrix $\mathbf{A}$ in the VAR model to be block diagonal, and therefore the CRP is guaranteed to hold for data generated from such a model. For convenience, we will refer to the second requirement as the "*Self-Reconstruction Property (SRP)*" from now on.

Definition 5.2.2 (Inradius [117])**.** *The inradius of a convex body $\mathcal{P}$, denoted by $r(\mathcal{P})$, is defined as the radius of the largest Euclidean ball inscribed in $\mathcal{P}$.*

By the definition, the radius of a $\mathcal{P}(\mathbf{X})$ measures the dispersion of the time series in $\mathbf{X}$. Naturally, well-dispersed data will yield a large inradius while data with skewed distribution will have a small inradius.

5.2.2 Theoretical Guarantees

One of our major contributions in this work is to provide theoretical guarantees for successfully recovering the clustering structure in the data.

Theorem 5.2.3. *Under the assumption of VAR model with a block diagonal transition matrix, we compactly denote $\mathcal{P}_0^l = \mathcal{P}(\mathbf{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l})$, $\mathcal{P}_1^l = \mathcal{P}((\mathbf{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l})$, $r_0^l = r(\mathcal{P}_0^l)$, $r_1^l = r(\mathcal{P}_1^l)$, and $r_0 r_1 = \min_l r_0^l r_1^l$ for $l = 1, 2, ..., k$, and let*

$$\rho = \frac{16\|\mathbf{\Sigma}\|_2 \max_j \Sigma_{jj}}{\min_j \Sigma_{jj}(1 - \|\mathbf{A}\|_2)} \sqrt{\frac{6\log d + 4}{T}}. \tag{5.2.1}$$

Furthermore, if

$$r_0 r_1 > \frac{\|\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}\|_{\infty,\infty} + 2\rho}{\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho}, \tag{5.2.2}$$

where $\boldsymbol{\gamma}_{\mathcal{S}_l} \in \mathbb{R}^{d_l}$ is a column of $\mathbf{\Sigma}_1$, then with probability at least $1 - 6d^{-1}$ the cluster recovery property holds for all the values of the regularization parameter λ in the range:

$$\frac{1}{r_0 r_1(\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho) - \rho} < \lambda < \frac{1}{\rho + \|\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}\|_{\infty,\infty}}, \tag{5.2.3}$$

which is guaranteed to be non-empty.

We defer the full proof of the theorem in the supplementary material. The theorem provides an upper bound and a lower bound for the regularization parameter λ, to successfully recover the underlying clustering structure in the time series: on the one hand, λ cannot be too large, otherwise $\mathbf{A}$ will be too dense to perform clustering on. On the other hand, as λ approximates 0 the connectivity among time series decreases because the optimal solution to (5.1.5) becomes more sparse. To guarantee the obtained solution is nontrivial (i.e., $\hat{\boldsymbol{\beta}}_i$ is nonzero), λ must be larger than a certain value. In addition, a lower bound on $r_0 r_1$ is established, which imposes a requirement on the dispersion of the covariance between time series within the same cluster. We further make the following remarks:

Remark 5.2.4 (Tolerance of Noise across Clusters)**.** *From* (5.2.2) *we see that, for the CRP to hold, the dispersion of the columns of $\mathbf{\Sigma}$ (each column is taken as a data point $\in \mathbb{R}^d$) needs to be sufficiently large. r_0 is the dispersion of covariance between time series in a cluster $\mathcal{S}_l$, and r_1 is the dispersion of lag-1 covariance between time series in a cluster $\mathcal{S}_l$. The RHS of* (5.2.2) *depends on the scale of $\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}$ and reflects the maximum correlation between the time series in one cluster $\mathcal{S}_l$ and any time series from all the other clusters.*

Remark 5.2.5 (Sample Complexity). *We can observe that the factor before the square root in* (5.2.1) *is bounded by the largest and smallest eigenvalue of $\mathbf{\Sigma}$, and therefore we can rewrite $\rho = \kappa\sqrt{(6\log d + 4)/T}$ where κ is a constant dependent on $\mathbf{\Sigma}$. We can further derive from* (5.2.2) *that $T > 4\kappa^2(6\log d + 4)(r_0 r_1 + 1)^2/(r_0 r_1 \|\gamma_{\mathcal{S}_l}\|_{\infty,\infty} - \|\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}\|_{\infty,\infty})^2$, which essentially indicates that the sample complexity for CRP to hold is $O(\log d)$. In other words, for the algorithm to succeed, the number of time series d is allowed to grow exponentially with the length of time series T; as long as $\log d$ is smaller than the length of time series T, our theory holds. Indeed, it is desired to see such a property for high-dimensional data, as d can often possibly far exceeds the number of samples T.*

Remark 5.2.6 (A Uniform Parameter λ). *Another direct observation from the main theorem is that we can find a uniform value for λ, within the range as specified in* (5.2.3)*, which can work for the regression task in* (5.1.5) *for all $i = 1, ..., d$. In other words, problem in* (5.1.5) *is solvable with a single λ and can be solved in parallel for each $i = 1, ..., d$.*

5.3 Evaluation

In this section, we demonstrate the correctness of our theoretical findings and the effectiveness of our proposed clustering algorithm on both synthetic and real-world data. We first experiment with different sets of parameters, including the number of time series d, the length of the time series T, and the number of clusters k in the data. The experimental results confirm that, when the required condition in Theorem 5.2.3 is satisfied, the clusters in the data can be recovered perfectly. We further apply the algorithm to a real-world data set where the task is to group sensor time series by their type of measurement (e.g., a temperature sensor vs. a humidity sensor). Our algorithm is able to outperform the state-of-art baselines by more than 20% measured by adjusted rand index.

5.3.1 Baselines

In our proposed clustering algorithm, we estimate the similarity matrix with the regularized Dantzig selector (referred to as CP). As baselines, instead of using our estimator, we consider the following methods to obtain the similarity matrix, and the rest of the clustering procedure remains the same as ours:

Correlation Coefficient (CC): In this baseline, we compute the Pearson correlation coefficient between all pairs of time series, and use these coefficients to construct the similarity matrix.

Cosine Similarity (Cosine): In the second baseline, we compute the pairwise cosine similarity for all the time series, and preserve only the similarity scores for the top-k nearest neighbors for each time series and put them as the row of the similarity matrix. Our experiment shows that the results are not sensitive to k and we set $k = 5$.

Autocorrelation (ACF): This baseline first computes the autocorrelation vectors (with a lag up to 50) for each time series, and then further calculates the Euclidean distance between each pair of time series based on the autocorrelation vectors. We use the implementation in [118] to obtain the distance matrix first, and then convert the distance into similarity score with Gaussian kernel function. (Smaller distances should map to larger similarity scores.)

Dynamic Time Warping (DTW): DTW is a popular method to compute the similarity between time series. Here we compute the pairwise DTW similarity score for all the time series, and then normalize similarity scores to between 0 and 1.

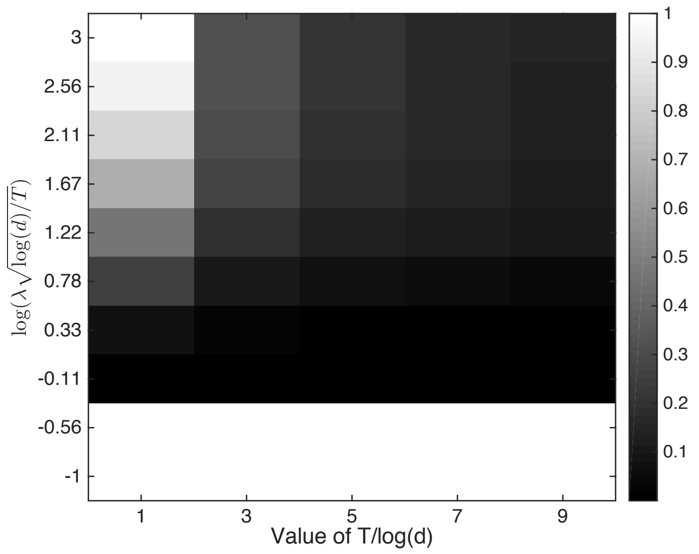
We also implement a baseline that does not rely on the similarity between time series:

Principal Component Analysis (PCA): In this method, PCA is first applied to reduce the dimensionality of each original time series by preserving d(=4) principle components, and then k-means is applied to these PCA scores for clustering.

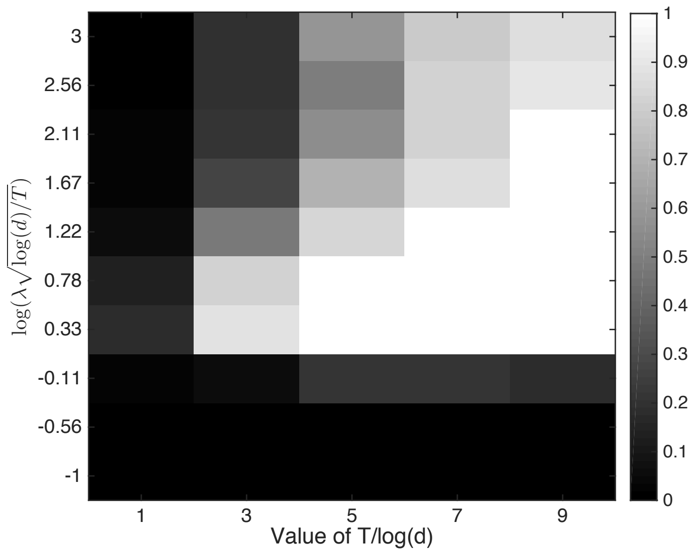
5.3.2 Synthetic Data

In this section, we show the effectiveness of our proposed clustering algorithm via numerical simulations. Particularly, the data $\mathbf{X}_t$ at a time point t are generated from a VAR model as defined in (5.1.1), and we generate the input time series $\mathbf{X}$ as follows: (1) We first generate the block diagonal transition matrix $\mathbf{A}$ with k clusters, and the values within each block are generated with a Bernoulli distribution; (2) Since we assume $(\mathbf{X})_{t=-\infty}^{\infty}$to be stationary, we then rescale $\mathbf{A}$ such that its spectral norm $\|\mathbf{A}\|_2 = \alpha < 1$; (3) Given $\mathbf{A}$, $\mathbf{\Sigma}$ is generated such that the elements on the diagonal equal to 1 and the off-diagonal elements are set to a same small value, e.g., 0.1. Then we rescale $\mathbf{\Sigma}$ to have its spectral norm satisfy $\|\mathbf{\Sigma}\|_2 = 2\|\mathbf{A}\|_2$; (4) Next, according to the stationary property, the covariance matrix of the additive noise $\mathbf{Z}_t$ follows $\mathbf{\Psi} = \mathbf{\Sigma} - \mathbf{A}^\top \mathbf{\Sigma} \mathbf{A}$, where $\mathbf{\Psi}$ must be a positive definite matrix; (5) We can then generate $\mathbf{X}_1$ from the multivariate normal distribution with the parameters generated in previous steps, and obtain the following $\mathbf{X}_t$ with the VAR model. We fix the number of time series d at 100, and choose the length of the time series T from a grid of $\{1, 3, 5, 7, 9\} \times \log(d)$ (rounded to the closet integer), i.e, the ratio of $T/\log(d)$ varies from 1 to 9. For each value of T, we repeat the data generation process for 100 times and report the average of the experimental results.

We first experiment with different values for the regularization parameter λ and examine if the two requirements as stated in Definition 5.2.1 are satisfied. We scan through an exponential space of λ from $1/(\log(d)/T) \times 10^{-1}$ to $1/(\log(d)/T) \times 10^3$ and define the metric *Self-Reconstruction Property Violation*



(a) Self-Reconstruction Property (SRP) violation rate for different $T/\log(d)$ against different λ with $k = 25$: too small a λ will produce trivial solutions ($\mathbf{A} = 0$, thus $VioRate = 1$) while a sufficiently large λ gives a solution satisfying both the nonzero and SRP requirements ($VioRate = 0$).



(b) Clustering quality for different $T/\log(d)$ against different λ with $k = 25$: cases satisfying the nonzero and SRP conditions yield perfect clustering results. It is also clear that the exact self-reconstruction condition ($VioRate = 0$) is not necessary for perfect clustering.

Figure 5.2: Self-Reconstruction Property Violation Rate and the Corresponding Clustering Quality (measured by Adjusted Rand Index) with Different $T/\log(d)$ Against Different λ.

Rate (VioRate) of the estimated transition matrix $\mathbf{A}$ as follows:

$$VioRate = \frac{\sum_{i,j \notin \mathcal{C}_l} |A_{ij}|}{\sum_{i,j \in \mathcal{C}_l} |A_{ij}|},$$

where $(i, j) \in \mathcal{C}_l$ denotes that the i-th time series $\mathbf{X}_{*i}$ and the j-th time series $\mathbf{X}_{*j}$ are in the same cluster $\mathcal{C}_l$ for some l (likewise for $(i, j) \notin \mathcal{C}_l$). By definition, $VioRate$ measures relatively how significant the predictive weights are for pairs of time series across different clusters, compared to the weights for pairs in the same cluster. For a trivial solution, i.e., $\mathbf{A} = \mathbf{0}$, the $VioRate$ is defined to be 1 while for a solution satisfying the self-reconstruction property, the $VioRate$ should be exactly 0. The violation rates for different $T/\log(d)$ and λ values when $k = 25$ are illustrated in Figure 5.2a; the results confirm our theoretical findings. We observe that when λ is small, the solution violates the nonzero requirement, thus the $VioRate$ being 1 (refer to the two rows at the bottom). When λ is sufficiently large within a range, the violation rates are zero, indicating all the entries in the off-diagonal blocks of the estimated $\mathbf{A}$ are zero, which satisfies the SRP. In Figure 5.2b, we show the quality of time series clustering (measured by adjusted rand index and higher is better) with the corresponding $\mathbf{A}$ obtained in Figure 5.2a. We can notice that cases perfectly satisfying the nonzero and SRP requirements can produce perfect clustering results. Furthermore, it is also clear that exact self-reconstruction condition is not necessary for perfect clustering.

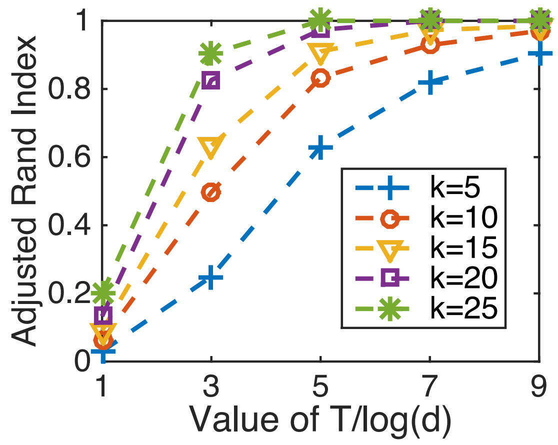


Figure 5.3: Clustering Quality for Different $T/\log(d)$ Against Different Number of Clusters k: larger k is better.

We next investigate how the number of clusters k affects the clustering performance, where we vary the value of k from 5 to 25. For the regularization parameter λ, we scan through the same exponential space as the above experiment with 5-fold cross-validation, and choose the one with the minimal cross-validation error. We fix $\|\mathbf{A}\|_2$ at 0.4 and report the average results of the 100 runs for each set of parameters as illustrated in Figure 5.3. We clearly see that a larger k leads to better clustering results, which makes sense since the more the number of clusters is, the sparser $\mathbf{A}$ is, and therefore the more accurate the estimation of $\mathbf{A}$ is.

We also examine the effect of the transition matrix's spectral norm $\|\mathbf{A}\|_2$ on the clustering quality. To this end, we set $\|\mathbf{A}\|_2 = \alpha$ and vary α from 0.1 to 0.9, and the covariance matrix $\mathbf{\Sigma}$ and $\mathbf{\Psi}$ are generated in the

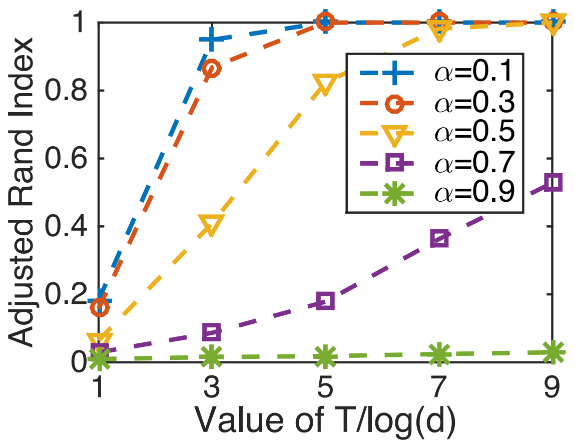


Figure 5.4: Clustering Quality for Different $T/\log(d)$ Against Different Values α for Spectral Norm $\|\mathbf{A}\|_2$: smaller α is better.

same way as described earlier. For the parameter λ, we take the same cross-validation procedure as above. We fix the number of clusters k at 25 and report the average results of the 100 runs for each set of parameters, as shown in Figure 5.4. We observe that, for a certain value of $T/\log(d)$, the clustering quality increases as the spectral norm of the transition matrix decreases. This indicates that the spectral norm of the transition matrix is a critical factor and verifies the theoretical findings in (5.2.1).

	CC	Cosine	ACF	DTW	PCA	CP
Synthetic Data-1 ($d = 50, T = 50$)	0.383 ± 0.171	0.521 ± 0.123	0.240 ± 0.147	0.282 ± 0.184	0.786 ± 0.139	0.943 ± 0.165
Synthetic Data-2 ($d = 50, T = 100$)	0.603 ± 0.181	0.551 ± 0.143	0.253 ± 0.109	0.410 ± 0.105	0.912 ± 0.141	1.000 ± 0.000
Real Data	0.617 ± 0.031	0.542 ± 0.014	0.362 ± 0.113	0.523 ± 0.119	0.456 ± 0.144	0.824 ± 0.025

Table 5.1: Experimental Comparisons with Baselines: results on synthetic and real data demonstrate the advantage of our proposed algorithm (CP), and each cell includes the average clustering performance (adjusted rand index) of 10 runs with standard deviation.

To compare our method with the baselines described in §5.3.1, we further conduct two sets of experiments on synthetic data with different parameters. To generate the synthetic data in the first experiment (referred to as Synthetic Data-1 in Table 5.1), we set the number of time series $d = 50$, the length of time series $T = 50$, the number of clusters $k = 5$, and the transition matrix's spectral norm $\|\mathbf{A}\|_2 = 0.5$. For the second experiment (Synthetic Data-2 in Table 5.1), we change T to 100, and the rest of parameters remain the same. We see that, when d is comparable to T in the first experiment, our method (CP) performs significantly better than the baselines. When the number of samples T is increased to 100, all the baselines see performance boost, while our method produces perfect clustering results.

One shall note the better performance of PCA, our understanding is that PCA extracts better explanatory components out of the sample covariance matrix, which still captures the underlying causal relationship between variables, though it does not consider the first-order temporal information as our proposed method does. For the other baselines, they simply compute similarity directly between variables, which is not sufficiently effective in characterizing the relationship between time series in the high-dimensional setting.

5.3.3 Real-world Data

To further examine how effective our proposed algorithm is in practice, we also apply it to a real-world data set, where the assumption of VAR model with block diagonal transition matrix might not be perfectly satisfied. The data set [19] contains data collected from 204 sensor time series from 51 rooms on 4 different floors of a large office building on a university campus. Each room is instrumented with 4 different types of sensors: a CO_2 sensor, a temperature sensor, a humidity sensor and a light sensor. The data from each sensor is recorded every 15 minutes and the data set contains one-week worth of data. There are missing values in the one-week period, so the total number of observations T is smaller than the number of sensor time series d. Our goal is to assign each sensor time series into the correct type cluster, e.g., a temperature cluster or a CO_2 cluster. Recognizing the type of sensors is often an important step for many useful applications. For instance, when applying analytics stacks comprised of a bundle of analytics jobs to a building for energy savings, every particular analytics job requires as input some specific types of sensors.

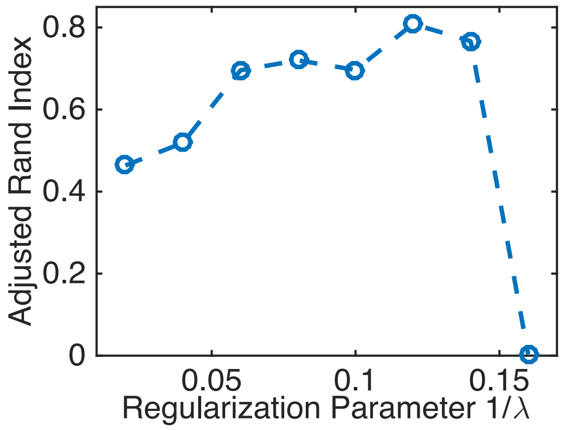


Figure 5.5: Clustering Quality of Our Regularized Dantzig Selector-based Spectral Clustering Algorithm: the algorithm works with a wide range of λ.

In this case, we do not know the values of the parameters in the sufficient condition in Theorem 5.2.3, so we cannot fine-tune λ. We roughly scan through the entire range of [0,1] for $1/\lambda$ and the results are shown in Figure 5.5 (the data points beyond 0.15 all drop to zero, thus omitted in the figure). It again confirms our theoretical findings in the sense that the proposed clustering algorithm can work when λ is sufficiently large, even not perfectly. We also examine how well the baselines (detailed in §5.3.1) perform on the real data set, and the results are summarized in Table 5.1. Our method can achieve more than 80% accuracy and outperforms the best baseline by more than 20%, indicating that our method can still be effective when the assumption of VAR with a block diagonal transition matrix might not be satisfied.

5.4 Summary

In this chapter, we describe a time series clustering method that relies on a new similarity measure in the high-dimensional regime, where the number of time series is much larger than the length of time series. Different from existing metrics, our similarity measure quantifies the "cross-predictability" between time series, i.e., the degree to which a future value in each time series is predicted by past values of the others. We impose a sparsity assumption and propose a regularized Dantzig selector estimator to learn the cross-predictability among time series for clustering. We further provide a theoretical proof that the proposed algorithm will successfully recover the clustering structure in the data with high probability under certain conditions. Experiments on both synthetic and real-world data verify the correctness of our findings, and demonstrate the effectiveness of the algorithm. For the real-world task of sensor type clustering, our method is able to outperform the state-of-art baselines by more than 20% with regard to the clustering quality.

Chapter 6

Relationship Inference

Previous chapters presented various techniques for inferring the sensor type. In this chapter, we describe a method for inferring another critical aspect of the sensor metadata — the functional relationships between equipment in the building. In particular, we propose a new approach to *automatically* identify the functional relationships between mechanical equipment. For example, the heating, ventilation, and air conditioning (HVAC) system in a commercial building typically contains 5-15 air handling units (AHUs) that heat or cool the air and circulate it to 200-500 variable air volume (VAV) terminal units. Each VAV contains dampers and heating coils to fine tune the volume and temperature of the air for a specific room, as illustrated in Figure 3.4. Thus, an HVAC system typically has thousands of highly-coupled mechanical parts and hundreds of controller modules, each acting independently. In current industry practice, one can query a building's BACnet network for the names of all these components, but not for the mechanical relationships between them. Therefore, when installing an analytics engine to a legacy or renovated building, obtaining these relationships is necessary. The methods proposed in this chapter will automatically determine how the VAVs and AHUs are connected to each other with minimal manual setup and configuration effort, so that an analytics engine can perform real-time monitoring and system-wide efficiency analysis, for example, model predictive control [119] or fault detection and diagnosis [9].

The research community has been trying to address the issue of inferring the relations among equipment and sensors [18, 136, 13], together with other physical aspects, including the type [16, 17, 146] and the location [19, 20]. However, all of these approaches involve a *manual* process that must be applied to each building individually, thus not scalable. For example, a recent study tries to infer the relation between VAVs and AHUs by perturbing the operation of each AHU and observing the response of VAVs [13]. However, this approach takes weeks to execute and requires knowledge of when and how to perturb operations in a way

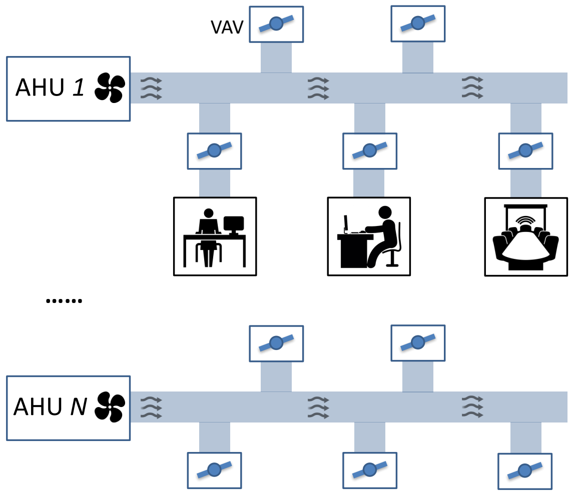


Figure 6.1: A typical building contains multiple air handling units (AHUs), each heating/cooling air and circulating it to dozens of variable air volume (VAV) boxes. Each VAV fine tunes the air flow supplied to a single room. Building analytics requires such information about the functional connection between equipment, which is currently performed manually in practice.

that does not interfere with building needs. Another study shows that contextual information can be extracted based on the names of data streams and equipment controllers [136]. However, this approach still requires manual assistance to interpret the naming convention of each building. Additionally, mechanical relationships are often not encoded in the equipment names and sometimes the names can even be missing entirely.

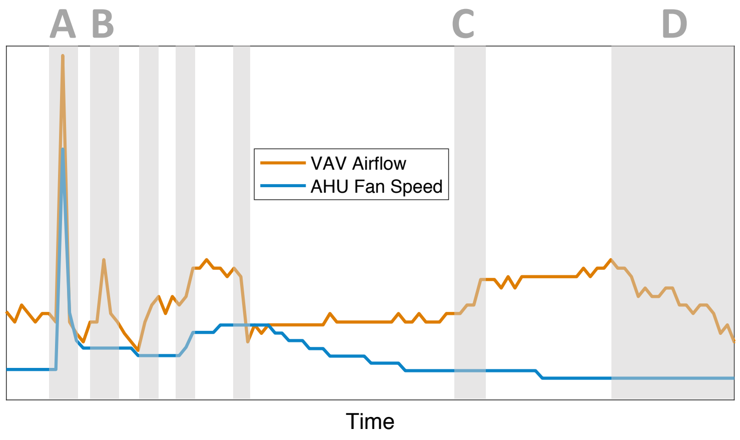


Figure 6.2: Example sensor readings from a VAV and its connected AHU. The VAV events (marked in grey) have many different causes. We desire a unified model that can automatically characterize normal operation for any data stream, and differentiate from other modes of operation.

The key intuition behind our solution is that functionally connected equipment is exposed to the same events in the physical world (e.g., a person entering a room or a supply air fan turning on) and thus will exhibit correlated events in their time series data. These events do not necessarily manifest as data outliers in the traditional sense, but rather as coordinated changes in the data progression; an event is a short subset of time series that may have exactly the same values as steady state operation but different noise levels, different rates of change, and/or different second derivatives. For example, Figure 6.2 shows example readings from a VAV air flow meter, where the events (marked in grey) can be outlier values (A and B), periods of gradual change (C and D) or discontinuities. Each of these events corresponds to a different state of the room, the equipment, or mechanically coupled equipment. We desire a unified model that can automatically characterize normal operation for any data stream, and differentiate from other modes of operation.

Because events are subsets of time series of indeterminate length, differentiating them from normal operation requires solving a combinatorial data segmentation problem — there are K possible states at each of the N timestamps and this gives a total of K^N possible combinations. To address these challenges, we take a probabilistic perspective of the events and develop a Markovian Event Model (MEMO) that uses a bank of kinematic models (each akin to a Kalman Filter [120]) to characterize the noise model and transition model for different modes of time series data. We introduce a latent variable to govern the switching between these models, reflecting the fact that the latent mechanical state underlying each data point is not observable. Here we assume that the equipment is in its normal mode of operation most of the time, and that important event types occur repeatedly and are statistically similar. Hence, using large raw data traces (e.g. one month), we can use a maximum likelihood estimator to jointly estimate a model of normal operation, common types of events, and the segmentation between them.

Once the events are detected, we identify relations between equipment by defining a correlation metric based on the inferred events in their time series data. The key challenge is that the data streams are affected by a mixture of latent factors, and not just the equipment relationships. For example, false correlations can be created by diurnal weather patterns, changes in room occupancy, the opening of a window, and other events. We filter these events and only look for co-incident events in both pieces of equipment. This dramatically reduces the number of spurious relations and helps to sift out pairs of equipment that are truly connected. Moreover, as we search for more of these correlated events over time, the probability of two pieces of equipment being correlated by random chance drops exponentially.

To evaluate our solution, we consider the functional relation between a particular pair of equipment in buildings discussed earlier: which VAV box is connected to which AHU. We evaluated the proposed approach on one-month data from 5 commercial buildings of various sizes which are geographically located across the US. Particularly, we performed evaluation in two different settings: 1) when we have knowledge about what type of sensors to use for the relation inference (e.g., only measuring the correlation between air flow volume in VAV and fan speed in AHU for the inference), our approach achieves 94.38% accuracy on average across all test buildings, compared to 85.49% by the best baseline; 2) when we do not have any knowledge about the type to use, i.e., we need to *automatically* select sensors for the analysis, our method achieves an average accuracy of 91.19%, compared to the possibly best accuracy of 95.64%. We believe this is a promising result, and expect our method to be generally applicable to broader domains of event detection and relation inference.

6.1 Methodology

The main insight behind our solution is that functionally connected equipment is exposed to the same events in the physical world, thus exhibiting correlated events in the attached sensors' readings. As we search for more of these correlated events over time, the probability of two streams being correlated by chance will drop exponentially and sift out the true relations. To this end, our approach first detects the latent events in sensor time series, and then identifies the relations among equipment based on the detected events, requiring minimal human interventions. Specifically, we employ a bank of kinematic models—each modeling one of the latent states—and a latent variable to govern the switch between these models. Once the events are detected, we use a correlation-based metric to further pinpoint the correlated set of events that are indicative of the equipment relations. In the rest of this section, we describe the basics of our continuous space Markovian Event Model, elaborate on how we estimate the latent states in the time series data, and finally explain the procedure of inferring the relation between equipment based on detected events.

6.1.1 The Markovian Event Model

In this study, we consider time series data from sensors, e.g., an air flow meter in a VAV that conditions some room, that are subject to changes in the physical world. As an example, Figure 6.2 shows the readings from an air flow meter of a VAV, where the events (marked in grey) have many different causes (labeled as A to D). The events can be outlier values (A and B), periods of gradual change (C and D) or discontinuities. There are also periods where the equipment stabilizes (the two long plateaus before and after C), although they stabilize around different values. Each of these periods corresponds to a different state of the room, the equipment, or mechanically coupled equipment. We desire a unified model that can automatically differentiate normal operation from other modes of operation in any data stream. A classical sequential model such as a basic Hidden Markov Model (HMM) would fail to model the underlying states with regard to such a requirement, because it simply models the probability of observing each value given the model parameters (e.g., mean and variance) of the possible states. In other words, HMM will interpret periods such as the two plateaus in Figure 6.2 as different states, since their statistics (e.g., mean) are apparently different.

To fulfill such a requirement, we model the latent "true" values of time series based on the observed values (i.e., we assume the observed time series is contaminated by noise), where we further assume each latent true value is inherently generated from an underlying latent state. Following the design, for the latent true values we assume the current true values are dependent on the previous ones, and the transition mode (e.g., how fast the transition is) would essentially reveal the underlying state. Such a design overcomes the inability of classical latent state models, which only center around the values, in differentiating various states in the data. Moreover, different from previous works, the parameters (e.g., variance) of each state in our model are learned globally and apply to all the data in time, thus allowing data that appear to be in different states (e.g., the two steady plateaus in Figure 6.2) to be modeled as in the same state.

Specifically, for the random variables in our proposed solution, we model both the value and the "velocity of change" in value for their time series readings. We consider the velocity of change because it helps to better characterize the time series in the case of changes. For example, when a person enters a room and the room temperature rises, the air conditioning system starts cooling and the sensor readings will exhibit sudden changes. A model that only models the value will not be able to closely track these sudden changes and results in significant residuals, i.e., modeling errors. Hence, we add the extra velocity dimension, so as to accommodate sudden changes.

Overall, there are three layers in the proposed model, as illustrated in Figure 6.3 (from top to bottom): (1) the latent state layer that governs the switch between states (e.g., steady or event) of the data; (2) the latent true values (e.g., the value of readings and the velocity of change in value) of the data; (3) the observed primitive readings of sensor data. Let $\mathbf{x}_t = [\mathbf{x}(t), \mathbf{v}_x(t)]^\top \in \mathbb{R}^2$ be the observed continuous readings from a particular sensor, e.g., a temperature sensor, where $\mathbf{x}(t)$ is the observed sensor value at timestamp t, and $\mathbf{v}_x(t)$ is the observed velocity of change at timestamp t, for $t \in \{1, 2, \ldots, N\}$. Likewise, we define the latent true values $\mathbf{y}_t = [\mathbf{y}(t), \mathbf{v}_y(t)]^\top \in \mathbb{R}^2$, where $\mathbf{y}(t), \mathbf{v}_y(t)$ are the latent true sensor value and the latent true velocity of change at each time t, respectively. For the latent discrete state variables $\mathbf{z}_{1:N}$, each $\mathbf{z}_t$ takes on a value from $\{1, 2, \ldots, K\}$, implying a total of K possible states in data at each time t.

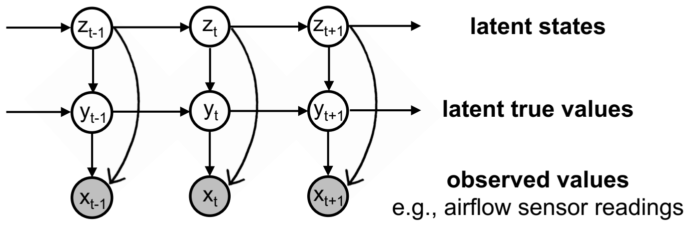


Figure 6.3: Graphical model representation of our Markovian Event Model.

Given the model structure specified in Figure 6.3, the joint distribution over the observed values $\mathbf{x}_{1:N}$, the latent true values $\mathbf{y}_{0:N}$, and the latent states $\mathbf{z}_{0:N}$ is given by:

$$\begin{aligned} &p(\mathbf{x}_{1:N}, \mathbf{y}_{0:N}, \mathbf{z}_{0:N}) = \\ &p(\mathbf{z}_0)p(\mathbf{y}_0|\mathbf{z}_0)\prod_{t=1}^{N} p(\mathbf{z}_t|\mathbf{z}_{t-1})p(\mathbf{y}_t|\mathbf{y}_{t-1}, \mathbf{z}_t)p(\mathbf{x}_t|\mathbf{y}_t, \mathbf{z}_t), \end{aligned} \tag{6.1.1}$$

where $\mathbf{z}_0$ and $\mathbf{y}_0$ stand for initial latent state and latent true values.

For the latent states $\mathbf{z}_{1:N}$, we assume they form a first-order Markov chain and the transition probability is given by:

$$p_{ij} = p(\mathbf{z}_t = j|\mathbf{z}_{t-1} = i), \quad 1 \leq i, j \leq K, \tag{6.1.2}$$

where K is the total number of possible states in the data. Thus, p_{ij} is modeled as a K-dimensional multinomial distribution. As we assume the observations are transformed from the latent true values subject to underlying noise, we assume the observed values are drawn from a Gaussian distribution centered at the latent true values. As a result, conditioned on the latent state $\mathbf{z}_t$, the probability of observing sensor reading data $\mathbf{x}_t$ is given by:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_t, \mathbf{z}_t) &= N(\mathbf{H}_{\mathbf{z}_t}\mathbf{y}_t, \mathbf{R}_{\mathbf{z}_t}) \qquad (6.1.3)\\ &\propto \exp\left[(\mathbf{H}_{\mathbf{z}_t}\mathbf{y}_t - \mathbf{x}_t)^\top \mathbf{R}_{\mathbf{z}_t}^{-1} (\mathbf{H}_{\mathbf{z}_t}\mathbf{y}_t - \mathbf{x}_t)\right] \\ &= \exp\left[\left(\mathbf{Y}_t - \mathbf{H}_{\mathbf{z}_t}^{-1}\mathbf{X}_t\right)^\top \left(\mathbf{H}_{\mathbf{z}_t}^{-1}\mathbf{R}_{\mathbf{z}_t}\left(\mathbf{H}_{\mathbf{z}_t}^{-1}\right)^\top\right)^{-1} \left(\mathbf{Y}_t - \mathbf{H}_{\mathbf{z}_t}^{-1}\mathbf{X}_t\right)\right], \end{aligned}$$

where $N(\cdot, \cdot)$ denotes a Gaussian distribution, $\mathbf{H}_{\mathbf{z}_t} \in \mathbb{R}^{2\times 2}$ is a matrix that linearly maps the hidden true values $\mathbf{Y}_t$ to the observation $\mathbf{X}_t$, and $\mathbf{R}_{\mathbf{z}_t} \in \mathbb{R}^{2\times 2}$ is the corresponding covariance matrix capturing the noise embedded in this transformation. For the latent true values $\mathbf{Y}_t$, we assume they also form a Markov chain and the transition probability follows Gaussian as well, i.e.,

$$\begin{aligned} p(\mathbf{Y}_t|\mathbf{Y}_{t-1}, \mathbf{z}_t) &= N(\mathbf{F}_{\mathbf{z}_t}\mathbf{Y}_{t-1}, \mathbf{Q}_{\mathbf{z}_t}) \qquad (6.1.4)\\ &\propto \exp\left[(\mathbf{Y}_t - \mathbf{F}_{\mathbf{z}_t}\mathbf{Y}_{t-1})^\top \mathbf{Q}_{\mathbf{z}_t}^{-1} (\mathbf{Y}_t - \mathbf{F}_{\mathbf{z}_t}\mathbf{Y}_{t-1})\right], \end{aligned}$$

where $\mathbf{F}_{\mathbf{z}_t} \in \mathbb{R}^{2\times 2}$ is the state transition matrix between $\mathbf{Y}_t$ and $\mathbf{Y}_{t-1}$, and $\mathbf{Q}_{\mathbf{z}_t} \in \mathbb{R}^{2\times 2}$ the covariance matrix capturing noise in the state transition.

6.1.2 Posterior Inference and Model Estimation

The key in applying the proposed model for event extraction is to infer the latent states $\mathbf{z}_{1:N}$ in a given time series $\mathbf{x}_{1:N}$. Based on our first order Markovian assumption in state transition, we develop an efficient sampling algorithm based on Gibbs sampling techniques [121] for posterior inference. The basic idea of Gibbs sampling is to accumulate posterior samples by sweeping through each variable to sample from its conditional distribution with the remaining variables fixed to their current values. Such sweeping is often performed iteratively until convergence.

Particularly, given our model structure in Figure 6.3, we can obtain the conditional probability of $\mathbf{Y}_t$:

$$\begin{aligned}&p(\mathbf{Y}_t|\mathbf{Y}_{t-1},\mathbf{Y}_{t+1},\mathbf{X}_t,\mathbf{z}_t)\\&\propto p(\mathbf{Y}_t|\mathbf{Y}_{t-1},\mathbf{z}_t)p(\mathbf{Y}_{t+1}|\mathbf{Y}_t,\mathbf{z}_{t+1})p(\mathbf{X}_t|\mathbf{Y}_t,\mathbf{z}_t),\end{aligned} \tag{6.1.5}$$

where the first term on the right-hand side is given in Eq. (6.1.4), the third term is given in Eq. (6.1.3), and the second term is computed as

$$\begin{aligned}&p(\mathbf{Y}_{t+1}|\mathbf{Y}_t)=N(\mathbf{F}_{\mathbf{z}_t}\mathbf{Y}_t,\mathbf{Q}_{\mathbf{z}_t}) \qquad (6.1.6)\\&\propto \exp\left[\left(\mathbf{F}_{\mathbf{z}_t}\mathbf{Y}_{t+1}-\mathbf{Y}_t\right)^\top\mathbf{Q}_{\mathbf{z}_t}^{-1}\left(\mathbf{F}_{\mathbf{z}_t}\mathbf{Y}_{t+1}-\mathbf{Y}_t\right)\right]\\&=\exp\left[\left(\mathbf{Y}_t-\mathbf{F}_{\mathbf{z}_t}^{-1}\mathbf{Y}_{t+1}\right)^\top\left(\mathbf{F}_{\mathbf{z}_t}^{-1}\mathbf{Q}_{\mathbf{z}_t}\left(\mathbf{F}_{\mathbf{z}_t}^{-1}\right)^\top\right)^{-1}\left(\mathbf{Y}_t-\mathbf{F}_{\mathbf{z}_t}^{-1}\mathbf{Y}_{t+1}\right)\right].\end{aligned}$$

We can hereby sample the latent variable $\mathbf{Y}_t$ from the composite Gaussian distribution, whose parameters are derived based on the above three Gaussian density functions. For sampling the latent state $\mathbf{z}_t$, we similarly have the conditional probability as follows:

$$\begin{aligned}&p(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{z}_{t+1},\mathbf{X}_t,\mathbf{Y}_t) \qquad (6.1.7)\\&\propto p_{\mathbf{z}_{t-1},\mathbf{z}_t}p_{\mathbf{z}_t,\mathbf{z}_{t+1}}p(\mathbf{X}_t|\mathbf{Y}_t,\mathbf{z}_t)p(\mathbf{Y}_t|\mathbf{Y}_{t-1},\mathbf{z}_t)\\&=p_{\mathbf{z}_{t-1},\mathbf{z}_t}p_{\mathbf{z}_t,\mathbf{z}_{t+1}}N(\mathbf{H}_{\mathbf{z}_t}\mathbf{Y}_t,\mathbf{R}_{\mathbf{z}_t})N(\mathbf{F}_{\mathbf{z}_t}\mathbf{Y}_{t-1},\mathbf{Q}_{\mathbf{z}_t}).\end{aligned}$$

We can then draw samples for each of the variables by sweeping through all the posterior conditionals. The theory of MCMC guarantees that the stationary distribution of the samples simulated under the Gibbs sampling algorithm is the target joint posterior that we are interested in [122]. Therefore, we run for a sufficient number of iterations of the sweeping process for each variable, and use the converged results to approximate the target posteriors.

The aforementioned posterior inference procedures depend on the availability model parameters, i.e., $\{\mathbf{F}_i, \mathbf{H}_i, \mathbf{Q}_i, \mathbf{R}_i\}_{i=1}^{K}$. In theory, we can estimate the parameters using Maximum Likelihood Estimation (MLE). However, as the sensor time series data in our problem reflects the physical property of the world, it is straightforward to assume the values (i.e., $\mathbf{X}_t$ and $\mathbf{Y}_t$) follow the physical law - sensor reading values at the next timestamp are based on the current one plus the product of velocity and time difference. As a result, for our problem, we fix $\mathbf{F} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ regardless of the value of $\mathbf{z}_t$, to reflect the physical law.

In order to estimate $\{\mathbf{Q}_i, \mathbf{R}_i\}_{i=1}^{K}$, we maximize the log data likelihood function defined by Eq. (6.1.1) based on the sampled results of $\mathbf{z}_{1:N}$ and $\mathbf{y}_{1:N}$. As the transition follows a Gaussian distribution, the maximization problem has a closed form solution; and for brevity, we only give the conclusion below:

$$\hat{\mathbf{Q}} = \frac{1}{N} \sum_t \left(\mathbf{Y}_t - \mathbf{F}\mathbf{Y}_{t-1}\right) \left(\mathbf{Y}_t - \mathbf{F}\mathbf{Y}_{t-1}\right)^{\top}. \tag{6.1.8}$$

Note that, we dropped the subscription $\mathbf{z}_t$ in $\mathbf{y}$ and $\mathbf{Q}$ to avoid cluttered notations. Specifically, when estimating $\mathbf{Q}$ for each state (i.e, $\mathbf{Q}_i$), one should only collect the inferred $\mathbf{y}_{1:N}$ values in this latent state. Similarly, we can estimate $\mathbf{R}$ with,

$$\hat{\mathbf{R}} = \frac{1}{N} \sum_t \left(\mathbf{X}_t - \mathbf{H}\mathbf{Y}_t\right) \left(\mathbf{X}_t - \mathbf{H}\mathbf{Y}_t\right)^{\top}. \tag{6.1.9}$$

We can estimate the transition probability p_{ij} for $\mathbf{Z}$ by collecting the sufficient statistics

$$p_{ij} = \frac{\sum_{\{t|\mathbf{z}_{t-1}=i, \mathbf{z}_t=j\}} 1}{N} \quad \text{for } 1 \leq i, j \leq K. \tag{6.1.10}$$

Basically, these quantities summarize the percentage of transitions from state i to j in the sampled $\mathbf{z}_{1:N}$ sequence.

Putting together the aforementioned posterior inference and parameter estimation procedures, we develop a stochastic Expectation-Maximization algorithm to iteratively refine the model parameters and the latent variable inference results:

E-Step: At each timestamp t, infer the latent variables $\mathbf{Y}_t, \mathbf{z}_t$ by Gibbs sampling based on the current model parameters (i.e., $\{\mathbf{Q}_i, \mathbf{R}_i\}_{i=1}^{K}$ and $\{p_i\}_{i=1}^{K}$) using Eq. (6.1.5)-(6.1.7).

M-Step: Update the model parameters by collecting the sufficient statistics of each latent variable, following Eq. (6.1.8)-(6.1.10).

We shall note that the sampling procedure for the latent variables in the E-step can be performed in time linearly with the length of time series N and further efficiently expedited via parallelism: because each variable depends only on the adjacent nodes before and after, we can update all the variables associated with even timestamps in parallel with all their neighbors at odd timestamps fixed, or vice versa. We alternate between these two groups and such parallelism helps us empirically achieve more than 30x speedup for the inference in our evaluation datasets.

6.1.3 State-dependent Transitions

In practice, when the data stabilizes within one state, it is reasonable to have the velocity of change follow the first-order model we specified earlier, i.e., the velocity at the current timestamp depends on the one at a previous timestamp. However, when the time series transits between states, e.g., from steady into events, or vice versa, a sudden change in velocity is expected. Still enforcing the first-order transition as in Eq. (6.1.4) will cause unnecessary early reaction or delay in latent velocity inference. In other words, the first-order dependency will enforce the velocity to start increasing ahead of events when transiting from steady into events (for example, from the first plateau to C in Figure 6.2), or stay decreasing post-events after exiting from events to steady.

To account for possible sudden changes in velocity across state transition boundaries, it is necessary to eliminate the dependency on its neighbors for velocity modeling in these boundary cases, i.e., the velocity at a timestamp such that at least one of its two adjacent readings is in a different latent state than its own. This can be intuitively understood as the system being perturbed by some sudden external stimulus, e.g., the AHU has been shut down. Basically, we have two different scenarios: 1) one of the adjacent readings is in a different state, and 2) both of the adjacent readings are in a different state.

For the first scenario, when modeling the posterior of $\mathbf{Y}_t$, we need to modify the form of its conditional probability by only preserving the emission probability and one direction of the transition probability. First, when $\mathbf{z}_{t+1}$ and $\mathbf{z}_t$ are different, we use the emission probability $p(\mathbf{X}_t|\mathbf{Y}_t)$ and the transition probability $p(\mathbf{Y}_t|\mathbf{Y}_{t-1})$, i.e., (note the difference below from Eq. (6.1.5))

$$p(\mathbf{Y}_t|\mathbf{Y}_{t-1}, \mathbf{Y}_{t+1}, \mathbf{X}_t, \mathbf{z}_t \neq z_{t+1}) \propto p(\mathbf{X}_t|\mathbf{Y}_t)p(\mathbf{Y}_t|\mathbf{Y}_{t-1}), \tag{6.1.11}$$

where the first term on the right-hand side is given in Eq. (6.1.3), and the second term is given in Eq. (6.1.4). We shall note that the above function only applies to the sampling for the velocity dimension (recall that $\mathbf{y}_t \in \mathbb{R}^2$). Due to the Gaussian assumption about the emission and transition probabilities, the posterior distribution in the transition case here can still be analytically derived as a composite Gaussian distribution.

When $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$ are different, to capture the unknown external stimulus that affect the sampling of $\mathbf{y}_t$, we introduce a global prior distribution of the latent true velocity $p(\mathbf{Y}_t|\theta_{\mathbf{z}_t})$ that is also state-dependent:

$$p(\mathbf{Y}_t|\mathbf{Y}_{t-1}, \mathbf{Y}_{t+1}, \mathbf{X}_t, \mathbf{z}_{t-1} \neq \mathbf{z}_t) \propto p(\mathbf{X}_t|\mathbf{Y}_t)p(\mathbf{Y}_{t+1}|\mathbf{Y}_t)p(\mathbf{Y}_t|\theta_{\mathbf{z}_t}). \tag{6.1.12}$$

For the second scenario, when modeling the posterior of $\mathbf{Y}_t$, we adjust the conditional probability to include only the emission probability, together with the prior probability, i.e.,

$$p(\mathbf{Y}_t|\cdot) \propto p(\mathbf{X}_t|\mathbf{Y}_t)p(\mathbf{Y}_t|\theta_{\mathbf{z}_t}). \tag{6.1.13}$$

6.1.4 Inferring Relations via Detected Events

The search space for equipment relation inference in a typical commerical building is huge. For a set of N sensors, the number of true relationships of interest is on the order of $O(kN)$, given the fact that each sensor is associated with k other sensors with respect to a particular relation; while the total number of pairwise relationships among them is on the order of $O(N^2)$. As we scale to thousands of sensors in a building, the total number of possible relations quadratically outnumbers the true relations of interest. To facilitate the inference process, our basic assumption for identifying the relations is that connected equipment will be exposed to the same events in the physical world, and thus will exhibit correlated changes in the data.

Yet, the key challenge is that the data streams are affected by a mixture of latent factors, and not just the equipment relationships. For example, false correlations can be created by diurnal weather patterns, changes in room occupancy, the opening of a window, and other external events. Thus, we need to filter these events and only look for co-incident events in both pieces of equipment. This can dramatically reduce the number of spurious relations and help to sift out pairs of equipment that are truly connected. Moreover, as we search for more correlated events over time, the probability of two sensors being correlated by random chance drops exponentially.

In our context, when there are events in an AHU, e.g., the outlet fan speed increases, these events will affect the downstream VAVs that connect to this AHU. Essentially, we should only look for the events in each VAV caused by the AHU for correlation, as they are the ones that reveal the true functional relation between them. Assuming that these resultant events in VAV would align temporally with its connected AHU events, we therefore filter spurious events by *masking* each VAV event sequence with AHU event timestamps—we only preserve the (detected) events in a VAV that align in time with the events in a given AHU, and reset the others to no events (or steady state), i.e.,

$$\mathbf{z}^{(i,j)}_{VAV}(t) = \mathbf{z}^{i}_{VAV}(t) \;\&\; \mathbf{z}^{j}_{AHU}(t) \text{ , for } t \in \{1, \ldots, N\}, \tag{6.1.14}$$

where $\mathbf{z}^{(i,j)}_{VAV}(t)$ is the value at time t in the binary event sequence of VAV i masked by AHU sequence j, $\mathbf{z}^{j}_{AHU}(t)$ is the value at time t in the event sequence of AHU j, and $\&$ denotes the bitwise AND operation. To be more specific, when correlating the event sequence from a VAV sensor i with each of the candidate AHU sensor j, we first mask the VAV event sequence with the given candidate AHU event sequence, and then calculate the cosine similarity sim_{ij} by

$$sim_{ij} = \frac{\sum_t \mathbf{z}^{(i,j)}_{VAV}(t)\; \mathbf{z}^{j}_{AHU}(t)}{\|\mathbf{z}^{(i,j)}_{VAV}\| \cdot \|\mathbf{z}^{j}_{AHU}\|}, \tag{6.1.15}$$

where $\| \cdot \|$ is the L-2 norm of a vector. After that, we assign each VAV to the AHU with which it has the highest similarity score. The masking step helps to further concentrate the comparison between VAV and AHU around periods when there are events in the AHU, which we believe drive the events in its connected VAVs and are the key to identifying the relations. Presumably, a masked VAV event sequence is expected to best correlate with its truly connected AHU among all the candidate AHUs. We will demonstrate in details the effect of the masking operation in the evaluation.

6.2 Evaluation

In this section, we empirically evaluate the effectiveness of the proposed solution in identifying the functional relation between equipment based on their sensor time series data. Both quantitative and qualitative evaluations are performed to demonstrate the inferred events and their value in relation inference.

6.2.1 Dataset

To evaluate our approach, we use the data from a set of 5 commercial buildings distributed across the US: the number of VAVs ranges from 100 to over 500, and the number of AHUs varies from 5 to 13. The ground truth for VAV to AHU association is obtained from our industrial partner that contracts with these buildings. The details of each building is summarized in Table 6.1. In the dataset, the typical number of sensing and control points attached to each equipment is between 6 and 13, and the smallest building in our set has over 1,300 points. The number of connected VAVs for each AHU falls in anywhere from a handful to more than 80. We shall note that, oftentimes, the connected VAVs to the same AHU can span over different floors within a

Building ID	10312	10320	10596	10606	10642
# of AHU	12	8	5	13	8
# of VAV	261	113	195	510	259
# of Floor	9	1	5	4	8
# of Sensor	2754	1318	2010	4811	2379
Size (SF)	300,000	65,000	150,000	490,000	181,397

Table 6.1: The details of the buildings used in our study, including the number of air handling units (AHU), variable air volume boxes (VAV), floors, and attached sensors.

building, which leaves the manual perturbation-based solution [13] inefficient, inaccurate, albeit considerably time-consuming. For the time series from each point, it is recorded every 15 minutes. And in each building we have collected one-month worth of time series data for all sensors.

6.2.2 Baselines

We compare our method with the following baselines, where each is used as an alternative method for event detection, and the subsequent relation inference step is the same as our approach.

Pearson Correlation Coefficient (CC): As the most straightforward baseline, we compute the pairwise Pearson Correlation Coefficient between each VAV and AHU, and then assign the VAV to the AHU with highest score.

Hidden Markov Model (HMM): For this baseline, we apply a K-state discrete HMM to infer the latent state in the sensor data.

Kalman Filter (KF): We compare against a standard KF that models the velocity of change in the time series data. We take the residues between the KF inferred true values and observed values as events for correlation analysis in the relation inference step.

Adaptive Event Detection (AED): AED models a time series by combining two Poisson distributions - one for the normal periodic component and the other for the rare event component [72]. We replace the Poisson distributions in the original method with Gaussians, to handle the continuous sensor readings in our problem.

Sliding Window Likelihood (SWL): We also compare our approach with a method that employs sliding windows and estimates the likelihood of having an event in each window sequentially [123].

Inspired by [124], we also compare our method with several baselines that combine the idea of time series clustering and bipartite matching. In particular, we first cluster the VAVs (using k-means with k set to the number of AHUs) and calculate the average linkage degree between every AHU and each group of VAVs. We then perform bipartite matching to assign each group of VAVs to one AHU, using the Ford-Fulkerson

Maximum Flow Algorithm [125]. The linkage degree is defined as

$$l(AHU_i, C_j) = \frac{1}{|C_j|} \sum_{k \in \{C_j\}} d(AHU_i, VAV_k), \tag{6.2.1}$$

where $C_j, \{C_j\}, |C_j|$ refer to the j-th cluster of VAVs, the VAVs assigned in the cluster, and the size of the cluster, respectively. And $d(a, b)$ calculates the cosine similarity between the two input streams, and we experiment with two different kinds of input streams—the primitive time series (TS) and the event sequence (ES). This gives us two baseline combinations. i.e., **KMeans-TS** and **KMeans-ES**.

6.2.3 Experimental Setup

For the number of possible states K in senor data, we set $K = 2$ by default, if not specified otherwise: we assume the time series data is either in a steady state or in an event state. For the first dimension of the input variable $\mathbf{X}$ to our algorithm(i.e., $\mathbf{X}(t)$), we simply use the observed sensor readings; and for the second dimension (the velocity, $\mathbf{v}_x(t)$), we take the difference between two successive readings in the observed sensor time series. For the initial values of the latent true values $\mathbf{Y}$ in our Gibbs sampling based posterior inference, we pass the raw sensor readings $\mathbf{X}$ through an Exponentially Weighted Moving Average filter (EWMA) [126] with a look-back window length (L = 5), and take the output as the initial values for $\mathbf{Y}$. For the latent state variable $\mathbf{Z}$, we assign random values from $\{0, 1\}$. Starting from the initial assignments, the sampled values in the first several iterations may not necessarily represent the actual posterior distribution; and thus, it is common to discard these samples. The discarded iterations are often referred to as the "burn-in" period, and we set the number of burn-in iterations $M_b = 20$. For the total number of samples M for each variable in Gibbs sampling, we set $M = 200$ and it is usually large enough to obtain converged results on our dataset. For the stopping criterion for the EM procedure, we set a threshold on the log data likelihood difference between two successive iterations, with a sufficiently small value.

6.2.4 Results & Analysis

Relation Inference Accuracy

As there are often multiple sensing points attached to each piece of equipment, we need to choose a pair of sensing points—one from VAV and one from AHU—to run our algorithm and compute the event sequence-based correlation score. In other words, domain knowledge about what type of points in different equipment are mostly correlated and best capture the relation becomes input of this problem, and typically such knowledge is acquired from a building manager; this is the *only* manual input required by our algorithm. In particular, in

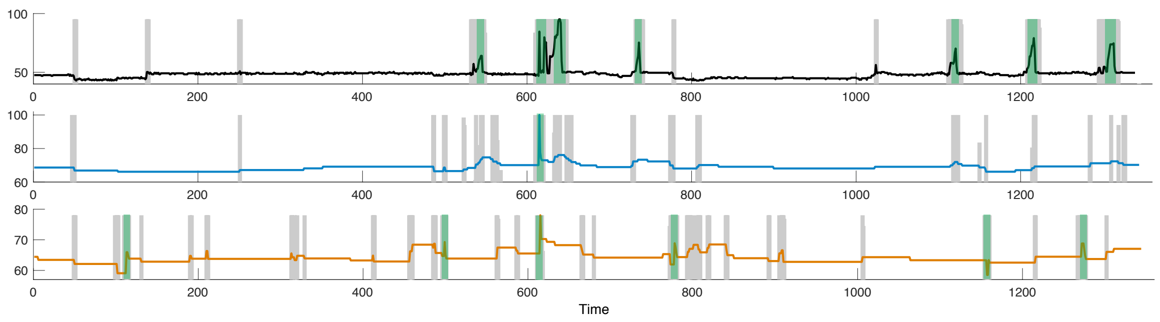


Figure 6.4: Examples of detected events for state number K=3 (events are marked with colored bars and the height of each bar indicates its probability) for a VAV (top), its associated AHU (mid), and a non-associated AHU (bottom). We see that events in different magnitudes are detected, i.e., dramatic ones (in green) and mild ones (in grey). The events in the VAV are clearly more correlated with the ones in its associated AHU.

Building ID	10312	10320	10596	10606	10642
CC	42.53	58.41	57.95	35.29	49.42
HMM	18.77	11.50	21.54	31.76	34.78
KF	90.04	80.53	87.69	80.29	88.91
AED	72.38	61.42	78.72	70.98	63.24
SWL	79.69	69.91	86.67	74.71	68.50
KMeans-TS	41.00	35.40	38.46	47.06	42.08
KMeans-ES	52.88	53.10	51.28	58.82	54.83
MEMO (K=4)	93.28	86.59	93.81	89.24	91.57
MEMO (K=3)	95.43	88.72	94.16	91.35	94.67
MEMO-NM (K=2)	96.17	84.96	95.38	89.41	94.59
MEMO (K=2)	**96.93**	**91.15**	**95.90**	**92.55**	**95.37**

Table 6.2: VAV assignment accuracy (%) by our method (MEMO, $K = 2$) against the baselines: Our algorithm consistently outperforms the baselines.

our experiments, we use `AirFlowVolume` in VAVs and `SupplyFanSpeed` in AHUs for evaluation, as the two are physically correlated. And later in the section, we also present a solution to automatically select a pair of points for this inference purpose.

We first pass each time series through our Markovian Event Model (MEMO) to obtain a sequence of detected events. Figure 6.4 shows examples of detected event sequences from the air flow measurement of a VAV, the fan speed measurement of the associated AHU, and that of a non-associated AHU. To illustrate the power of MEMO in detecting events, we set $K = 3$ in this case study, i.e., the model identifies three states. We label them as steady state, mild event state (in grey bars), and dramatic event state (in green bars). Our MEMO is able to detect the significant events, while ignoring lots of relatively strong noise (such as the small change around timestamp 100). By visually comparing the event sequences detected among the three pieces of equipment, we are able to recognize that the VAV event sequence is more closely aligned with the event sequence from the associated AHU, than the one from the non-associated AHU.

Formally, the assignment *accuracy* is defined as the proportion of VAVs that are correctly assigned to the associated AHU in a building. The main results are summarized in Table 6.2. On average, our algorithm achieves 94.38% accuracy across all the buildings, compared to 85.49% by the best baseline. First, we see that the clustering+bipartite matching baselines (KMeans-TS/ES) could not produce competitive results as our method, in large because the clustering step generates spurious groups of VAVs in the first place, thus the significant matching errors. Yet, we do see an average improvement of $\sim$14% by KMeans-ES over KMeans-TS, where the former performs clustering over the event sequence, which we believe makes better sense than using the primitive time series.

For the event-based baselines, as expected, HMM does not perform well because it simply models the state at each point based on its raw sensor reading value, thus producing almost random state assignments. KF performs the closest to ours, as it also considers the velocity of change in the data, yet it only encodes the influence of past observations on the current values, but not the influence from future data (as standard KF only performs forward computation of expected values). For AED, because it relies on a normal periodic component to model the steady states, e.g., daily patterns, any slight deviations from the norm, such as stronger cooling due to temperature rise on a hotter day in our data, will be modeled as events, thus resulting in too many false positive events. In contrast with these methods, our algorithm complements the strength of a kinematic model by taking into account both the backward temporal information, i.e., the influence of future data, and learns the state parameters that globally apply to data across all time.

Overall, we attribute the better performance of our algorithm to its more accurate event detection, which produces less spurious candidates to compare against. The results imply that our algorithm could have profound impacts: with only limited knowledge about what type of sensors to use for running the analysis, the algorithm is able to *automatically* identify the correct relations for about 94% of the equipment. Our solution can significantly reduce the manual effort required in the relation identification process, as the current state-of-the-art industrial solution stands. One should note that, due to the absence of the event ground truth for the dataset, we were unable to directly evaluate the event detection accuracy. However, here we indirectly demonstrate the accuracy of event detection with a proxy, namely, the relation inference accuracy.

Effect of Masking

Since we believe only the events in a VAV caused by its associated AHU help reveal the functional relation between them, we take the masking operation when performing the proposed correlation analysis, i.e., to preserve only the events in a VAV that overlap in time against a given AHU. To demonstrate the effect of masking, we present an illustrative example in Figure 6.5 where from top to bottom are an AHU, its connected VAV, and a non-connected VAV (we set $K = 2$ to simplify the illustration). We can find that, after masking the

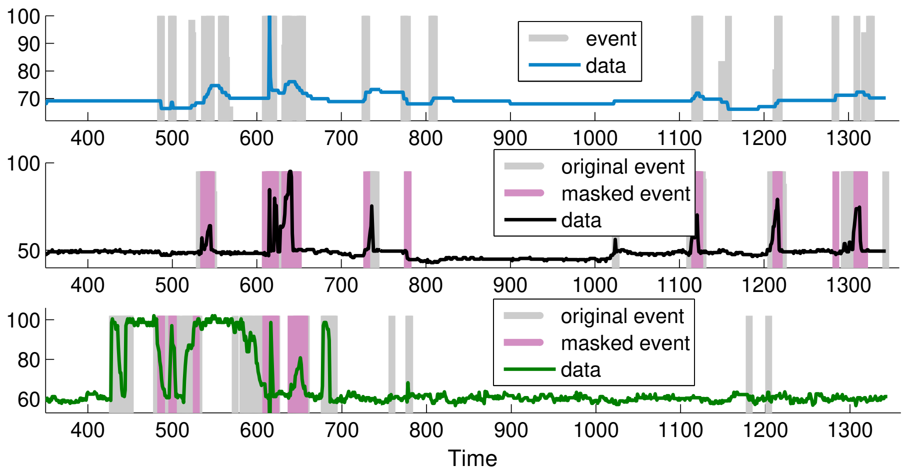


Figure 6.5: Event sequences from an AHU (top), a connected VAV (mid), and a non-connected VAV (bottom): preserving VAV events that overlap in time with AHU events (marked in pink overlaid on grey) helps to better pinpoint the correlated set of events, for inferring the relations.

original events (in grey), the remaining events (in pink) in the connected VAV (mid) clearly better correlate with the AHU (top), than the non-connected VAV (bottom) does, which fails to correlate in the later part of sequence). To be more specific, before and after the masking operation, the similarity score is 0.73 vs 0.87 between the top and mid, while the score is 0.59 vs 0.31 between the top and the bottom; masking clearly helps to concentrate on the potentially correlated set of events. We also modify the correlation step in the original method to only compare the primitive event sequences between VAV and AHU without the masking operation, i.e., we compute

$$sim_{ij} = \frac{\sum_t \mathbf{z}^i_{VAV}(t)\, \mathbf{z}^j_{AHU}(t)}{\|\mathbf{z}^i_{VAV}\| \cdot \|\mathbf{z}^j_{AHU}\|}, \tag{6.2.2}$$

and assign a VAV to the AHU with the highest correlation score. The results are listed in Table 6.2 (MEMO-NM), and we observe clear degradation, especially for building 10320 and 10606.

Upon closer inspection, we note that the VAV placement in these two buildings is significantly denser than the others — for these two buildings there are more than 100 VAVs on each floor, whereas for the others the number is between 20-30. The denser VAV placement in these two buildings means that each individual conditioning zone is essentially smaller, and thus more subject to the thermal influences from other nearby zones, generating more changes in the VAV data stream. As a result, after adding the masking operation to each VAV's event sequence, which is designed to filter out irrelevant events with respect to an AHU, our algorithm is able to better reveal the functional connections among the equipment.

Effect of Velocity Independence

We also examine the effect of relaxing dependency in velocity modeling at the transition periods, as explained in Section 6.1.3. To this end, we instead preserve the dependency between velocity during transition periods and inspect the inferred events and velocity. We see from Figure 6.6, the velocity reacts early or delays when dramatic changes occur, resulting in false positive events (marked in green). We conclude that adding independence into velocity modeling with respect to the latent state change benefits the event detection.

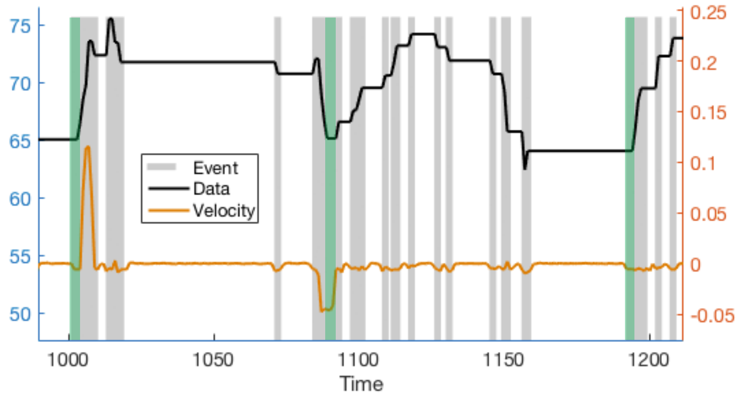


Figure 6.6: Without the independence constraint, the modeled velocity results in early reactions and delays, consequently producing false positive events (marked in green).

Effect of Number of States

For the set of results above, we set the number of states in the event detection model $K = 2$, i.e., we assume there is either an event or none. Here we further inspect how the choice of K affects the relation inference performance. Particularly, we increased K to 3 and 4 (see MEMO ($K = 3, 4$) in Table 6.2) and found that choosing a larger K did decrease the relation inference accuracy. When the number of states increases, the types of inferred events become inconsistent across streams (see Figure 6.4); a more rapid event in one sequence might look like at a relatively mild event in another, or vice versa. In other words, the alignment between different types of identified states is not guaranteed in our model, as the model parameters (i.e., $\{\mathbf{Q}_i, \mathbf{R}_i\}_{i=1}^{K}$) are not shared across sequences. We need to emphasize that as $\{\mathbf{Q}_i, \mathbf{R}_i\}_{i=1}^{K}$ specify the inherent noise in modeling the time series data from different pieces of equipment, it is not reasonable to share them across streams, which will enforce the same noise distribution on all streams.

Effect of Amount of Data

To examine the effect of the amount of training data on the inference accuracy, we vary the amount of data passed to our proposed algorithm from 1 week to 8 weeks. From Figure 6.7, we find that, as we increase the amount of time series data for model training, both our model (MEMO) and the best baseline improve in the inference accuracy and plateau after reaching around 4 to 5 weeks of data.

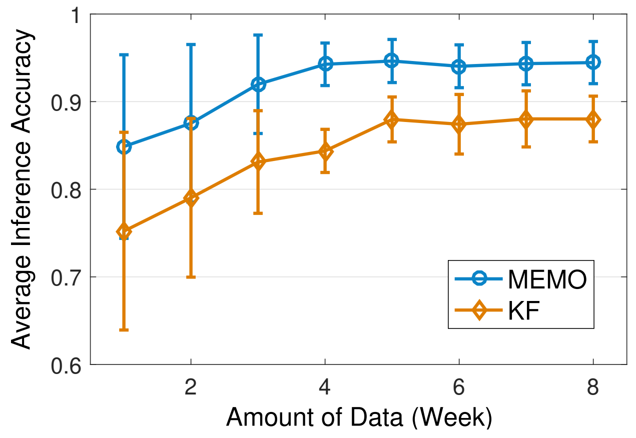


Figure 6.7: The effect of amount of data on average inference accuracy across all the test buildings: the more the data, the higher the accuracy of our solution (MEMO). Same trend is observed also for the best baseline (KF).

This also verifies our intuition that, as we search for more correlated events between points over a longer time period, the probability of two points being correlated by random chance drops, thus resulting in higher accuracy of relation inference. We also observe a similar trend for the best baseline (KF), yet its performance is still worse than our method.

6.2.5 Fully Automation for General Applicability

A premise in the previous evaluation is that we are given the knowledge about which points to use for correlation and that pair of points are consistently available across all buildings. However, such a premise might break in practice, as the type of measurements available could vary from site to site, namely, in different buildings. For example, Table 6.3 presents the VAV assignment accuracy using different pairs of points in one of our test buildings. We see that the accuracy of using different pairs of points varies drastically, and furthermore, the set of points available in another building could be much different. Consequently, we need a strategy to automatically decide which pair of points to use, in order to make it generally applicable across sites, or even problem areas. The question raised here is, "*Can we automatically decide which pair of points to use to best identify the relations?*" As a futuristic direction, we make a preliminary attempt at automating the selection of points to use for identifying the relations.

Intuitively, the points representative of the relations are expected to yield a "deviant" correlation score for the pair of truly associated equipment, compared to the other candidates. In other word, ideally, the correlation score between the pair of equipment that are truly associated should as much separate from the rest of pairs

VAV \AHU	Supply AirPressure	Supply AirTemp	Supply FanSpeed
AirFlowVolume	75.22	33.63	91.15
DischargeAirTemp	57.52	42.48	36.28
SpaceTemp	13.27	15.04	31.86

Table 6.3: VAV assignment accuracy (%) by using different pairs of points in one of the buildings: the performance varies drastically from pair to pair.

as possible. Following such an intuition, we design a metric to measure the deviation of the highest score produced by a particular pair of points. Given the m-th point from the i-th VAV and the n-th point from the j-th AHU, we obtain the similarity scores between that particular VAV and all candidate AHUs:

$$s^{m,n} = \left\{sim_{i,j}^{m,n}\right\}, \text{ for } 1 \leq j \leq L, \tag{6.2.3}$$

where $sim_{i,j}^{m,n}$ stands for the correlation score between the i-th VAV with the j-th AHU using the m-th sensing point in VAV and the n-th sensing point in AHU, and j sweeps through all L AHU candidates in a given building. Then we measure the following quantity for each pair of points:

$$p(s^{m,n}) = \sigma(s^{m,n}) \times \Big(1 - N\big(s_{\text{max}}^{m,n}|\mu(s^{m,n} \setminus s_{\text{max}}^{m,n}), \sigma(s^{m,n} \setminus s_{\text{max}}^{m,n})\big)\Big),$$

where the notation "\" denotes set difference, and $N(v|\mu, \sigma)$ gives the likelihood of observing v under the Gaussian distribution parameterized by the mean μ and standard deviation σ. Then for each VAV, we decide a pair of points to use by $\text{argmax}_{m,n} p(s^{m,n})$, and the rest of the assignment process remains the same as the original approach.

Building ID	10312	10320	10596	10606	10642
Oracle	96.55	91.15	97.95	92.55	100.00
AutoPair	95.79	84.88	89.72	85.98	99.60

Table 6.4: Assignment accuracy using automatically selected pair of sensors (AutoPair) for correlation against the oracle accuracy.

To evaluate the approach, we first need the oracle accuracy of using automatically chosen pair. To this end, we calculate the relation inference accuracy enumerating all pairs of points, such as presented in Table 6.3, and pick the highest accuracy achieved among all pairs as the oracle accuracy. This can be understood as the upper bound accuracy of our algorithm in this dataset. In Table 6.4, we summarize the accuracy using the pairs decided with our proposed metric, compared with the oracle accuracy. On average, while the oracle hits 95.64%, our approach achieves 91.19%, which degrades only slightly compared to the 94.38% accuracy

Algorithm 4: Genetic Algorithm-based Co-location Inference

```
Input: Raw sensor readings of $K$ typesOutput: Room assignment of each sensor
1. Obtain the event sequence for each sensor.
2. Randomly assign each sensor into a group for every type $i$ and repeat $N$ times to generate $N$ initial
  candidates.
3. Calculate all the pairwise Pearson correlation scores between sensors in each group using their
  event sequences, and obtain the total sum $S_{corr}$ over all groups.
4. Create a new generation of $N$ assignment candidates, where each is generated by swapping $M$
  random pairs of sensors of the same type $i$, and calculate the correlation score sum $S_{corr}$ for every
  new candidate.
5. Rank the $N$ candidates by their correlation score sum $S_{corr}$ and replace the bottom $N_{low}$ candidates
  with the top $N_{low}$ candidates.
6. Go back to 4 and repeat until $S_{corr}$ converges for the top candidate on the ranking.
7. Output the room assignment according to the top candidate.
```

when given the pair of points to use a priori. We believe our approach is promising in automating the relation inference process and making it generally applicable across sites, or even domains.

6.2.6 Co-location Inference

Our intuition behind the solution to the equipment connection inference problem is that, as two pieces of equipment are connected, we shall observe correlated changes in their sensor readings. As a brief demonstration next, we shall note that the same intuition can also be applied to automatically identify sensors in the same room, i.e., the co-location relationships. Extracting such co-location information is critical to applications that monitor and control specific rooms, e.g., to optimize occupant's comfort, and identifying the co-located sensors is not a one-time effort due to retrofitting or renovation of a building [19].

As sensors are placed in the same room or physical space, they are again subject to the same events, e.g., a person entering the room or sun coming up, and therefore we shall also observe coincident changes in these sensors' data. Specifically, a sensor should best correlate with the others in the same room, rather than others that are not co-located. To solve the co-location inference problem, we again leverage the detected events in each sensor time series data as in identifying the functional relationships, and search for a grouping among sensors such that it maximizes the total sum of the intra-group pairwise correlation between sensors for all groups. However, obtaining such a grouping needs to solve a combinatorial problem with huge search space, and we exploit a genetic algorithm (GA) [127] to facilitate the search process. The whole algorithm is described in Algorithm 4. We shall note that we assume the type information of each senor and we assume there is only one sensor for each type. As a result, the mutation operation in Step 4 of the algorithm only happens within the same type between two rooms.

We evaluate the above algorithm on one-week data collected from the sensors in an office building, where

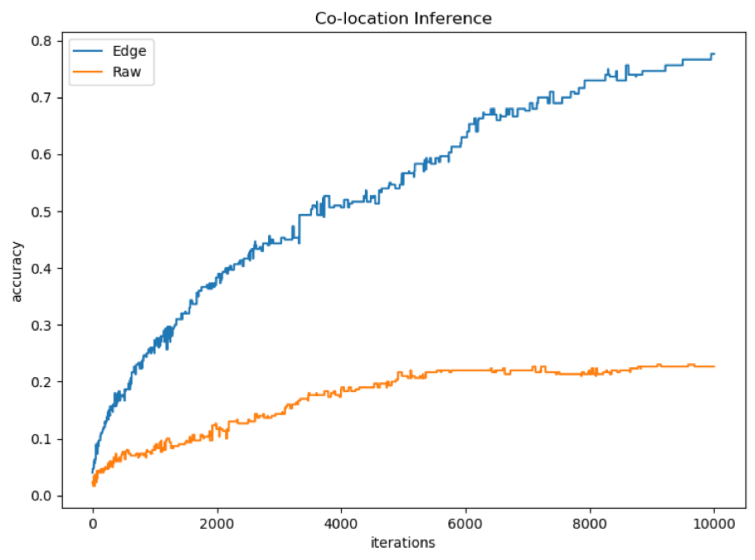


Figure 6.8: Co-location Inference Accuracy using the detected events and Genetic Algorithm.

in total there are 51 rooms. Each room has four types of sensors: a CO2 sensor, a humidity sensor, a light sensor, and a temperature sensor. The data from these sensors is reported to an sMAP [96] archiver over 6LowPAN [128] every 15 seconds. As baselines, we run the GA-based algorithm on the raw time series data and the subcomponent used in our preliminary study [19] of each sensor. The results are summarized in Figure 6.8. We see that the event-based case clearly outperforms the others, which corroborates our statement that looking for coincident events can also help to identify co-located sensors.

6.3 Summary

In this chapter, we tackle the problem of inferring another important contextual aspect of the equipment in buildings, apart from the sensor type — how equipment is functionally connected with another. We first develop a Markovian Event Model to identify the latent events in the time series data of equipment, and then further propose a correlation-based metric to locate the correlated set of events for inferring the underlying relations among equipment. We evaluated our approach on data from 5 commercial buildings and the approach achieves 94.38% accuracy, compared to the 85.49% by the best baseline. We also made an attempt to fully automate the solution and we only observe slight performance degradation. As a brief demonstration, we also present the results on using correlated events to identify co-located sensors. We believe the solution is promising and can be generally applied to event detection for times series data and relation inference in broader domains.

Chapter 7

An Integration, Benchmark, and Development Framework for Metadata Inference Methods

To fully realize the potential of smart building applications, we would need a system that is able to quickly discover the points in a building, interpret their data context, and express it in a *standardized* and *uniform* way. Doing so would require a common metadata schema for buildings.Typically, a building metadata schema defines and provides a structure for describing and representing the resources in the building. The representation would comprise two kinds of information about each point in the building — the type and the relationship. An example conforming to such a representation would read as *a temperature sensor is in room 501*, which contains a first entity with the type being temperature measurement, a second entity with the type being room, and the relation between these two entities, i.e., A `is in` B. In the same spirit, Brick — a recently proposed schema by the research community — is designed to improve over and complement existing industrial building metadata schemata (e.g., Haystack [129], IFC [130], and SAREF ontology [131]) with better expressibility, extensibility, and usability [132, 133]. Brick particularly provides a full hierarchy of entity classes as TagSets and a systematic way of describing various relationships among entities. Brick enables portable building applications built upon common vocabularies of classes and relationships to find required entities, instead of adapting to each individual target building's convention.

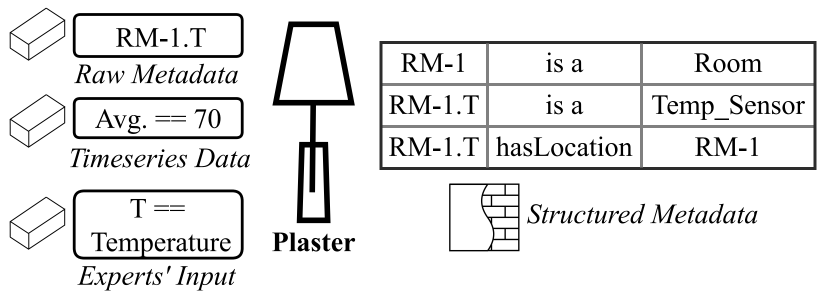
However, currently converting (existing) metadata to a schema such as Brick still requires tremendous manual effort, and we need a more usable, accessible solution for non-technical users such as building managers to close the loop. Along with the techniques discussed in this dissertation, various methods have been proposed to partially automate the inference of sensor context [134, 161, 136, 137, 17, 146, 13, 138, 139], with each focusing on different aspects of the contextual information. Some methods recognize all entities in raw metadata [134, 136, 138], including the site name, floor and room identifiers, and point type. Other methods identify only the point type based on either the raw metadata [137, 161], timeseries data [17], or both [146]. Still other methods focus on inferring relations between entities, including the spatial relationships [19, 20] and functional relationships [139, 13]. In order to reduce the manual effort required, these methods either only exploit the information available within each individual building [134, 136, 137, 161, 17] or they apply "transfer learning" from one building to the next [146, 138]. Importantly, while all of these prior works exploit the common attributes of each point — the alphanumeric text-based metadata and/or the numerical time series readings, they differ significantly with regard to the inference scope, input/output format and structure, algorithm interface, evaluation metric, etc [140]. Consequently, the resultant lack of compatibility among the combination of methods precludes the possibility of combining and comparing them effectively and systematically. There is still no standalone, versatile solution so far.

We need a unified framework that is modular and extensible, enabling both existing techniques to be explored, as well as new algorithms and techniques to be developed and tested, to advance the state of the art in building metadata normalization. Our vision is to build a framework akin to the Scikit-learn of the building metadata normalization problem. To this end, this chapter we introduce Plaster with the following contributions:

- We design and implement Plaster, a unified, flexible, extensible, and modular framework that incorporates existing metadata normalization methods, along with a set of data models, evaluation metrics and canonical functionalities commonly found in the literature. Altogether these enable the integration of different methods into a generic workflow as well as development and evaluation of new algorithms.

- We present the first systematic evaluation of state-of-the-art methods via a set of unified metrics as well as building datasets. Our evaluation covers a wide spectrum of aspects, such as how accurate each method is in inferring the same kind of label, how many different kinds of labels each method can produce, and how many human labels are required to achieve certain performance. The experiment results reveal that there is no one-size-fits-all solution and properly combining them would produce better results. This evaluation would not have been possible without Plaster, given the heterogeneous algorithms, inputs, and datasets used in earlier work.

- We discuss a number of future research directions inspired by our results. We believe Plaster provides a comprehensive framework for further development of new algorithms, techniques and workflows for



RM-1	is a	Room
RM-1.T	is a	Temp_Sensor
RM-1.T	hasLocation	RM-1

Figure 7.1: To facilitate the deployment of portable smart building applications, Plastercollects the state-of-the-art metadata normalization methods and provides a standardized way for users to map unstructured metadata to the Brick format. Plasteralso provides a standard benchmark for comparing different methods and spurs new ones.

metadata normalization, as well as mapping them to a structured ontology like Brick, enabling seamless smart buildings applications in the future.

7.1 Background and Related Work

In this section, we first present the background of building metadata normalization. Next we describe the prior work, highlighting our specific contributions beyond the state of the art.

7.1.1 Building Metadata Schema: Brick

Without metadata represented in a unified, standardized building-agnostic schema, deploying a smart building application requires adapting it to each target building's naming convention. Thus, the existence and adoption of a standardized metadata schema directly affect the cost of deploying smart building applications [141]. Indeed, there already exist several metadata schemata such as Industry Foundation Classes (IFC) [130] and Project Haystack [129]. However, as they have incomplete vocabularies and cannot fully describe the relationships required by common building applications [142], Brick has been introduced as a complete, extensible, flexible, and usable metadata schema for application portability [132, 133]. Brick comprises a full hierarchy of classes (Fig. 7.2a) and covers a canonical set of relationships between entities (Fig. 7.2b). The classes in Brick are also referred to as TagSets as they consist of multiple Tags. For example, `Temperature` and `Sensor` are Tags constituting a TagSet, `Temperature Sensor`. With Brick, one can instantiate the classes to represent actual entities (e.g., a sensor or a room) and relate an entity to another via a particular relationship. The table in Fig. 7.1 presents an example of a temperature sensor using Brick: the original raw metadata `RM-1.T` is mapped to an instance of `Temperature Sensor`; and to represent relational information such as its location, one can explicitly associate it with other entities such as room-1, which is

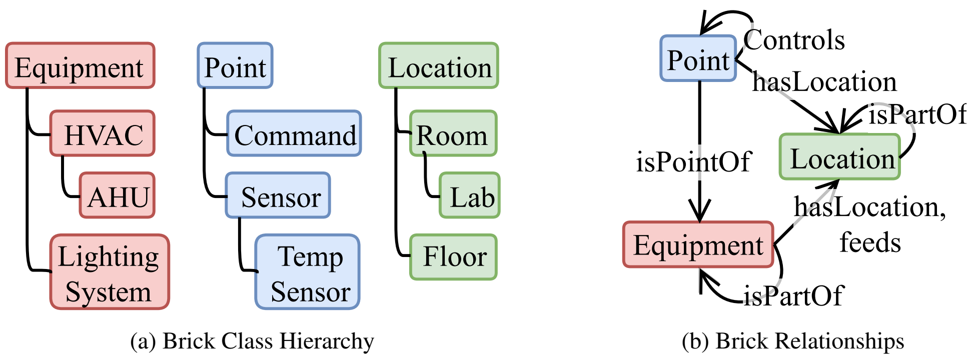


Figure 7.2: Brick comprises (a) a full hierarchy of classes and covers (b) a canonical set of relationships between entities required by common smart building applications.

again an instance of type `Room`. With Brick, a user can avoid using custom tags to describe both the entity type and its relationships with others, which fundamentally makes running portable applications across buildings feasible. Therefore, we choose Brick as the target mapping convention in this chapter as it is capable of representing (almost all) the resources and relationships that are needed in smart buildings, and is in our opinion more comprehensive compared to other schemata.

7.1.2 Metadata Inference Methodologies

We identify three dimensions of variance in existing metadata inference methods: 1) the type of data sources exploited, 2) the kinds of labels covered, and 3) the degree of human input required.

First, there are three different types of **data sources** we can exploit in buildings — raw alphanumeric metadata in existing building management systems (BMSes), numerical historical timeseries data for points, and control perturbations. The raw **metadata** in BMSes, also referred to as *point names*, usually encode various kinds of information about the control and sensing points, including the type of sensor, floor and room numbers, HVAC equipment ID, etc. The metadata for each point is typically a concatenation of an arbitrary number of abbreviations and each describes a different aspect of the point, following certain naming convention. The metadata within a building often exhibits strong learnable patterns, and various works have leveraged such patterns for metadata inference [134, 135, 136, 137, 138]. However, the pattern or structure in the metadata varies significantly across buildings, and thus the pattern extracted from metadata often does not generalize from one building to another. Secondly, modern BMSes also collect historical **time series** readings of each point in the building, which contain information that indirectly reveals what the point is and its relationship with others. For example, the range of the readings can indicate the type of sensor (e.g., readings around 70 units are likely for a temperature measurement while readings around 500-1,000 units are more likely for a CO2 sensor), and the correlation between different streams can indicate the relationship (e.g., observing correlated changes in the readings might indicate they are monitoring related events). Works that leverage the characteristics of timeseries data include [19, 20, 17]. In addition to the raw metadata and timeseries data, one may also perform **controlled perturbation** in a building, in order to identify attributes such as functional relationships across different entities [139, 13]. Controlled perturbations are manually injected to the building, e.g., to manually turn off an air handling unit, and can create new patterns in operations that produce unusual or unique changes in the timeseries data, but require more careful and sophisticated designs.

Existing metadata inference methods focus on producing two **kinds of labels** — following the definition in Brick — entity types and relationships between entities. The entity type refers to the type of measurement of a point and there is a wide variety in its possible set of labels, while the relationships include how points are connected to each other, whether they are in the same room/zone, etc. A few methods infer all the available information (e.g., both kinds of labels) encoded in the raw metadata [134, 136, 138], whereas many others identify the point type only [161, 17, 146, 143], which is the most important aspect of a point in buildings, or infer the relationships only [19, 20, 13, 139].

While different methods all aim to reduce the amount of manual effort in normalizing metadata, the degree of **human input** (i.e., manual labeling) required by each of them varies from fully supervised to semi-supervised to completely unsupervised. Particularly, supervision, or human input, in this context is the annotation or labels that a human expert provides to interpret the point for its type, location, relationship with others, etc. Supervised learning has been used to learn the point types based on timeseries data or raw metadata, where both clean, accurate labels from experts [17], and crowd-sourced labels from occupants in the building [143] have been explored in the literature. For the set of semi-supervised solutions, active learning is a key direction [144] and leads to the reduction in the amount of human labels. Specifically, an active learning algorithm iteratively selects a most informative example and queries an expert for its label to improve a model for normalizing the metadata [135, 136, 137, 138]. On the other hand, transfer learning method [145] has been developed to exploit information from existing buildings and completely eliminate human effort from the process of inferring metadata structure in a target building without any training examples [146]. Similarly, Scrabble [138] is another method that exploits existing buildings' normalized metadata, though through an active learning procedure. Table 7.1 summarizes these various methods with regard to the above criteria. In this work we show that, while each of these techniques has its advantages, our proposed *meta*-framework — Plaster— can leverage and integrate these different techniques in a complementary manner to yield much better results.

Additionally, although not being directly related to the metadata inference problem, there are frameworks in other domains that integrate different algorithms and create composable workflows, including general machine

Table 7.1: State-of-the-art metadata normalization methods produce various types of labels using different data sources. They also employ different machine learning (ML) algorithms involving diverse types of user interaction.

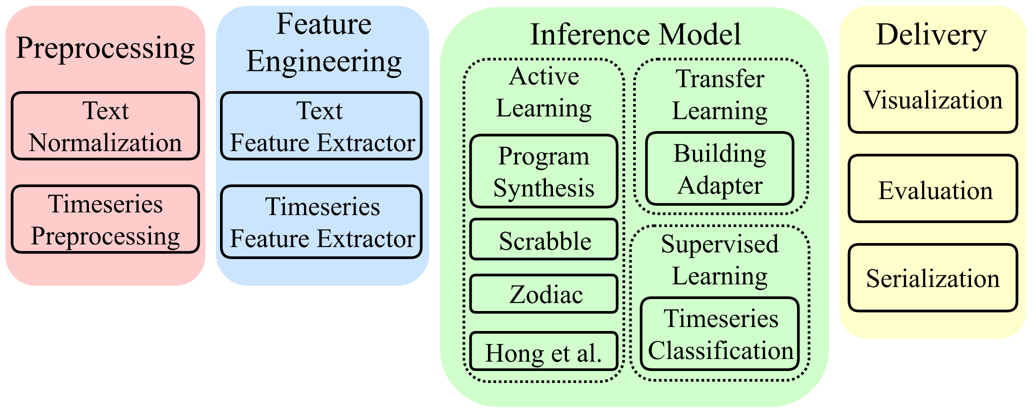
Method	**Label Produced**	**Data Source**	**ML**
Bhattacharya et al. [136] Scrabble [138]	All entities	Raw metadata	AL
Zodiac [137] Hong et al. [135]	Point type	Raw metadata	AL
Fürst et al. [143]	Point type	Raw metadata	CS
Building Adapater [146]	Point type	Raw metadata, Timeseries	TL
Gao et al.[17]	Point type	Timeseries	SL
Hong et al.[147]	Point type	Timeseries	UL
Pritoni et al. [13] Quiver [139]	Functional Relationship	System Perturbation, Point Label	UL

AL: Active Learning TL: Transfer Learning
SL: Supervised Learning CS: Crowd Sourcing
UL: Unsupervised Learning

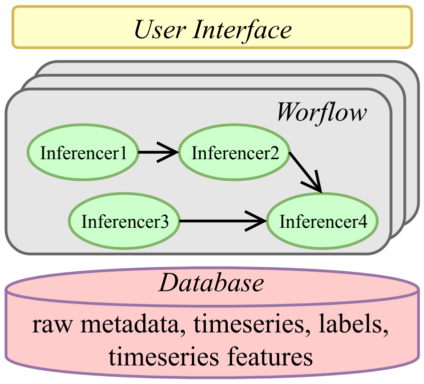
learning analytics [148], recommendation systems [149, 150], non-intrusive load monitoring [151, 152]. To the best of our knowledge, Plasteris the first framework of its kind that enables the exploration and integration of various algorithms on building metadata inference, as well as provides the ability to systematically compare related algorithms and results.

7.2 Plaster Framework

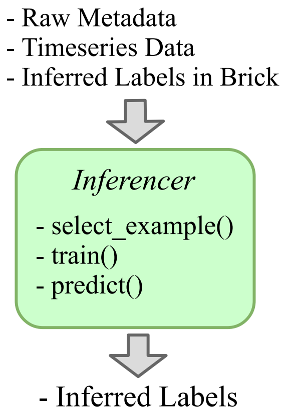
Plaster delivers an integration and development framework, as well as a standard benchmark, for different metadata inference methods, despite the heterogeneity of the data model, learning scope, interaction style, and evaluation metrics of each method. To tackle the heterogeneity, we propose a **modular** design with two levels of abstractions by identifying the commonalities among existing methods. As the *first* level of abstraction, Plasterviews a metadata inference method as an ensemble of a key **inferencer** and several other **reusable** components. These components provide a variety of functionalities and each component defines its own functions and programming interfaces. This way, we provide users with the flexibility in choosing the data model, learning scope, and inference algorithm as needed. As the *second* level of abstraction, an inferencer, which is the core component, contains multiple common functions that we identify by summarizing existing metadata inference solutions. Because of the modular design of Plaster, it also enables and facilitates the invention of new **workflow** where a user can connect different inferencers to essentially create a new algorithm, without changing or re-implementing prior algorithms. In the rest of this section, we will explain the architecture of Plaster, elaborate the core idea of inferencerthat is the key to enabling the creation of workflows, and also describe the other components of Plaster.



(a) Components in Plaster



(b) Workflow



(c) inferencerInterface

Figure 7.3: PlasterArchitecture: Plasteradopts a modular design and incorporates a variety of components, among which the core is a family of inference algorithms. Each algorithm is abstracted as an ensemble of common functions, which allow the communication between different algorithms. Such a design enables not only flexible invention of a workflow composed of any algorithms, but also systematical comparison between different algorithms.

7.2.1 Architecture

In Plaster, we abstract each method as an ensemble of components, and overall there are four basic categories of components as illustrated in Fig. 7.3a (from left to right): preprocessing, feature engineering, inference models, and results delivery functions.

The *preprocessing* component includes standard functions such as denoising and outlier removal for timeseries data, and lowercasing and punctuation removal for textual metadata, via an interface to utilize existing libraries such as SciPy and Pandas. There are also database (DB) I/O functions for both the metadata and timeseries data. We use universally unique identifiers to identify points and one can access both the textual metadata and timeseries data through the identifiers. For the timeseries DB functions, Plasterbuilds upon an open-source library [153] piggybacked on MongoDB, which is dedicated and optimized for timeseries data operations on large data chunks. For *feature engineering*, there are a number of existing libraries, such as the most widely used scikit-learn [154] and a recent effort – tsfresh [155]. However, none exists as customized for the timeseries data from buildings, considering their uniqueness such as the distinct diurnal patterns. Hence, we incorporate and extend the feature sets[1] implemented by Gao et al. [156], which contain various feature functions customized for building timeseries data. In addition to the original feature sets, we provide straightforward programming interfaces for a user to select a subset of features out of these predefined features and a lightweight yet effective feature selection function based on lasso [157]. We shall demonstrate the effectiveness of the feature selector in Section 7.3.4. For text features, we provide an interface for Bag-of-Words [85], sentence2vec [158], and LSTM-based auto-encoder [159]. *Delivery* components consist of the set of evaluation metrics (detailed in Section 7.3.1), user interaction mechanisms to provide additional supervision for inferencerupdate as needed, and serialization tools that convert the inference results into the Brick format (e.g, triples and graphs).

7.2.2 Inferencer

At the core of Plasteris a collection of state-of-the-art metadata inference methods. We examine these algorithms and identify similar procedures among them. We therefore abstract these procedures as a series of common functions, encapsulate each as a parameterized interface, and formulate a standardized way of constructing an inference algorithm. We use an abstract class – inferencer– to represent an algorithm (e.g., Scrabble, Building Adapter, etc), which maintains its own model for metadata normalization under these abstract interfaces, together with other standardized input and output interfaces. Such abstraction decouples the complex procedures in individual algorithm and allows new algorithms to be easily included into the framework.

At a high level, an inference algorithm in the building metadata domain aims to achieve the best possible accuracy with the largest coverage using the minimal set of labeled examples. Therefore, an inferencertypically contains a few steps: 1) the algorithm selects as training set the most "informative" example(s) based on its own criterion and acquires the labels for the selected example(s) from a human expert; 2) the model updates its parameters based on the latest training set after the new examples are added in the previous step, and then 3) the model predicts all types of labels (e.g., point type, location, relationships, etc.) covered by the algorithm. Plasterabstracts each of the above steps as a function, viz, `select_examples()`, `train()`, and `predict()`, respectively, as shown in Fig. 7.3c; and we design an inferencerto be a composition of these functions. We shall note that, although these functions appear to be only able to compose an active

[1]We refer the readers to their original paper [156] for more details on each feature set.

learning-based procedure, we design the `select_examples()` function to be generic enough such that any fully to semi-supervised learning algorithm can fit into this template. When obtaining examples for a supervised or transfer learning algorithm, the `select_examples()` function simply includes all the labeled or transferred examples for training at one time, rather than being iteratively done as in active learning. For active learning, these steps are repeated in iterations involving a human expert to best learn the model, while for a supervised learning or transfer learning algorithm, these steps are mostly executed just once with already labeled examples.

We also define standardized input/output interfaces for these common functions to enable the communication between different inferencers, which permits the creation of workflow as we shall discuss shortly. For inputs, an inferenceraccepts three types of sources: raw metadata, timeseries data, and the corresponding labels of examples. We provide a wrapper to digest two types of raw metadata commonly found in existing systems: 1) point names accessible through vendor-given interfaces that are widely used in the literature, and 2) metadata in BACnet [160] including entries such as `BACnet Description` and `BACnet Unit`. Timeseries data is stored as a series of timestamped values and the data for each specific point is associated with a unique identifier of the point for indexing and future retrieving. As each inferencercan learn and produce various types of labels as discussed in Section 7.1.2, an inferencer can take three kinds of labels at different granularities: point type labels, labels for all entities existing in the metadata, and character-level parsing with BIO tagging [162].

The ultimate goal of each individual inferenceris to generate structured metadata. Since Brick is capable of representing different kinds of metadata such as the types of entities and the relationships between them, we express the `predict()` method's outputs of each inferencerfollowing the Brick's format. Particularly, the outputs are a list of triples for entities and relationships in a building as explained in Section 7.1.1. Consequently, an inferenceris capable of representing different inference results in the same format. For example, Zodiac [137] infers the point types, which can be represented as "X is a Y" triples, while Quiver [139] infers the co-location relationship for multiple sensors expressed as "X1 hasLocation Y" and "X2 hasLocation Y". Such different types of inference are serialized in the same format of Turtle [163] using the vocabulary in Brick.

Additionally, for each inferencerwe include the confidence of its inference results produced by the original algorithm in its output so that an inferenceris able to more flexibly sift through and use another inferencer's results. Specifically, we store the confidence for each produced triple within the inferencer. However, the notion of confidence is unique per inference algorithm with different meanings. For example, Support Vector Machine's confidence is usually measured by the distance to the hyperplane, while in Naïve Bayes, it is the probability of observing the example given the model's parameters. Thus, we restrict the interpretation of confidence score within each individual inferencerdespite the values of those metrics being uniformly normalized to be between zero and one. We shall show how this is useful in real workflows in Section 7.3.3.

As a natural outcome, Plaster provides a standard benchmark for different metadata normalization algorithms. With the unified interfaces in inferencer, to do so is straightforward as one only needs to specify the set of algorithms he/she wants to compare as well as the type(s) of data to ingest and designate a building for the comparison. We will demonstrate with concrete examples in Section 7.3.2.

7.2.3 Workflow

The standardized interfaces in inferenceralso enable the creation of a workflow for metadata normalization. A workflow is a hybrid method comprised of multiple algorithms, each being an inferencerin Plaster, where the output of an inferenceris passed to another while each inferencerexecutes its inference procedure independently. While a single inferencerusually only infers one aspect of the metadata, a workflow would potentially be able to infer multiple or all the aspects of the metadata by employing different inferencers. Each inferencermay have a different learning objective as described in Section 7.1.2, and Plasterhelps to systematically leverage the advantages of each. For example, Building Adapter [**?**] (BA) is a transfer learning based algorithm that infers point types without any human inputs, but usually with a potential low recall. Instead of starting from scratch, the output of BA can constitute an initial training set for Zodiac [137] to jump start its learning procedure and potentially reduce the amount of manual labels required. Various use cases of workflow enabled by Plasterare elaborated and evaluated in Section 7.3.3.

When executing, the workflow function call will invoke the corresponding functions (i.e., `select_examples()`, `train()`, and `predict()`) in each of its inferencers in the order specified in the workflow, with an additional connecting step that obtains and applies the previous inferencer's prediction results to the next. The process of applying a precedent inferencer's results vary across different inferencers so that a human integrator should specify how to digest such predictions inside the inferencer's methods. If some of the inferred relationships by the previous inferencerare less confident, the next inferencershould filter the results or simply avoid using them. On the contrary, if the previous inferencer's inference is more confident than the current inferencer's, it can discard its own inference and adopt the previous one. In the example of connecting BA and Zodiac, Zodiac would need to be able to select only the prediction results with high confidence from BA and subsequently add them into its own training set inside `select_examples()`. Although it is an additional implementation over the base algorithms, we shall see such synergy could lead to better results.

7.3 Evaluation

In this section, we demonstrate how Plasterconveniently enables systematic comparisons of different metadata inference algorithms, the creation of new workflows by connecting multiple algorithms, and new functionality such as feature selection. We also present our evaluation results and discuss about the implications of the results.

7.3.1 Experimental Setup

Datasets

Building	Vendor	Year	Size (ft^2)	# Points	# Point Types
A-1	Johnson Controls	2004	150,000	4,594	108
A-2	Johnson Controls	2004	150,000	4,357	111
B-1	Trane	2011	100,000	1,300	60
C-1	Johnson Controls	2009	141,000	2,300	31
D-1	Automated Logic	2009	217,000	8,292	147

Table 7.2: Case Study Buildings Information.

We obtain a subset of the study buildings used in Brick [132], which consists of five buildings from four different campuses, including the raw metadata and timeseries data for about a month. Table 7.2 summarizes the details of each test building. While this collection of five buildings are not comprehensive for building metadata research, we argue that they are representative enough with regard to the diversity in vendors, sizes, years of construction, etc. For building D1, the original author did not release the timeseries data, and therefore, one shall notice that later D1 is not included in evaluations that involve timeseries data.

Evaluation Metrics

Overall, we consider three aspects supported by Plaster when evaluating each algorithm:

- *Inference Accuracy*: How accurate are the predictions of an algorithm in terms of its original learning purpose?
- *Inference Coverage*: What kinds of labels can an algorithm infer?
- *Human Efforts*: How many examples does an expert need to provide in the learning process of an algorithm?

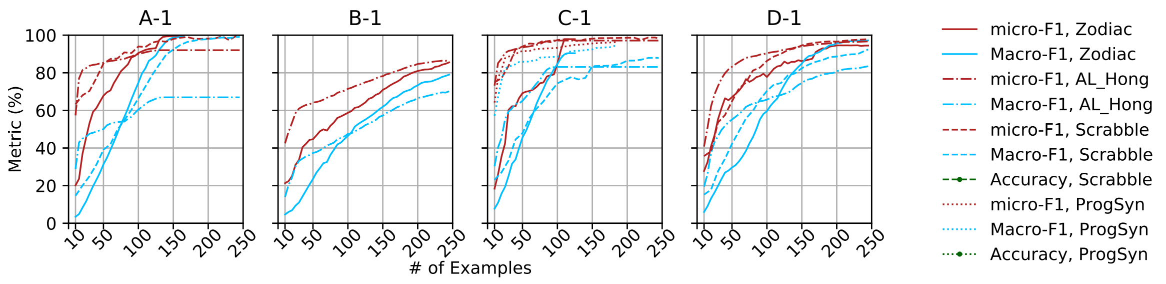
In this study, each algorithm infers one or multiple kinds of labels for a point. For example, Zodiac infers only one kind of labels, which is the point type, whereas Scrabble also identifies other kinds of labels such as location aside from the point type. For each kind of label, every possible Brick tagset is treated as a class (e.g., for point type we have room temperature, supply air temperature, etc), and we evaluate the inference performance considering all the kind(s) of labels each algorithm produces. To measure how accurate the inference results are for an algorithm, we calculate the Micro-averaged F1 (MicroF1), Macro-averaged F1 (MacroF1), and example-level accuracy. MicroF1 globally counts the total true positives, false negatives and false positives regardless of the class, while MacroF1 calculates the same quantities for each class and then finds their unweighted mean. MacroF1 indicates how many different classes can be correctly inferred, which is an important metric for a building dataset that typically has an (extremely) imbalanced class distribution, with the points related to heating and cooling in domination. For example, while `Zone Temperature Sensor`s might frequently exist in HVAC systems, specialized points such as `Gas Meter`s are generally rare. For *example-level accuracy*, it is defined as the ratio of the number of correctly labeled examples over the total number of examples. Specifically, an examples is considered to be correctly labeled if and only all kinds of labels existing for it are correctly inferred. We use this metric along with the F1 scores when an algorithm produces more than one kind of labels.

We measure human efforts by the number of examples labeled by an expert during the model learning process. For point type inference, an example is usually a mere point type label given the raw metadata of the point. For the examples used for inferring all possible entities, they contain more information aside from the point type label, such as equipment ID and location. Although the amount of information in the examples are different, we consider the effort for labeling an example to be the same because the required knowledge per example is similar.

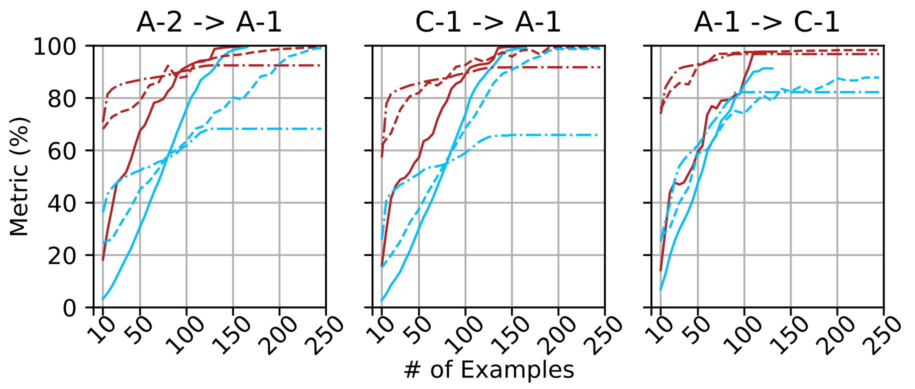
Inferencers Included in Plaster

We have refactored and incorporated the following algorithms into Plaster: Hong Active Learning [135] (referred to as AL_Hong hereafter), Bhattacharya et al. [136] (referred to as ProgSyn), Zodiac [137], Building Adapter [146], and Scrabble [138]. We exclude algorithms from the evaluation that require the actual control of actuations in buildings [139, 13] because such experiments are not practical in most buildings. However, they fit into Plasterwell as part of a workflow in the real world such as building commissioning. Plasteris open-sourced[2] and implemented in Python, which has identified itself as one of the most popular programming languages for scientific computing for its high-level programming interfaces, interactive nature, and vibrant ecosystem of supporting libraries such as Numpy and Pandas. To maximize reproducibility while minimizing development time, we deliberately reuse the original implementation of each method and modularize them according to the interface design in our framework.

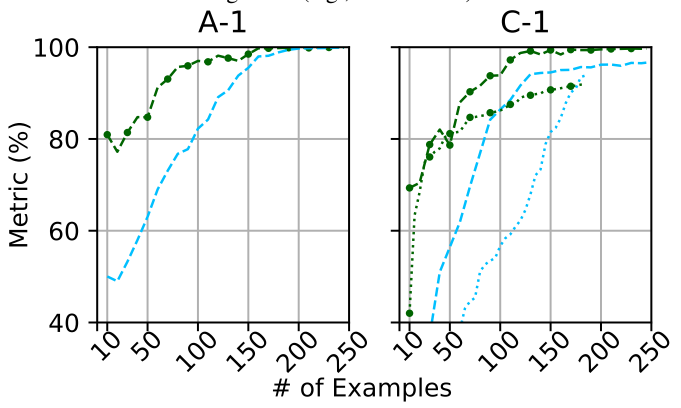
[2]The repository is publicly available, including its source code, running examples, and accompanying datasets.



(a) Learning rate for inferring point type by different algorithms on 4 buildings starting from scratch (i.e., zero training set).



(b) Learning rate for inferring point type, exploiting existing building's normalized metadata. X $\rightarrow$ Y indicates applying X's normalized metadata to initialize the learning for Y (e.g., A-1 $\rightarrow$ C-1).



(c) Learning rate for inferring all entities in the raw metadata from scratch.

Figure 7.4: Comparisons of Different Algorithms on Various Buildings: (atop each figure) The alphabet represents a campus and the number represents a building on that campus (e.g., A-1). We leave out Scrabble's results for B-1 and ProgSyn's results for A-1, B-1 and D-1 due to the limited types of labels in these buildings. All experiments are averaged over four runs and the legend is shared across all figures.

7.3.2 Benchmarking

Enabled by the design of workflow, Plasterallows a user to easily select an inferencerand specify the type of input ingested, the test building to use, and the evaluation metric; this fundamentally facilitates fair and systematical comparisons of different metadata inference algorithms, i.e., *benchmarking*. We focus on three different scenarios and next present the results for each.

Active Learning for Point Type Inference

In this scenario, we evaluate a set of active learning-based algorithms for their learning efficiency in inferring point types, which is the most important aspect of building metadata. We include two algorithms that exclusively work for this purpose — AL_Hong [135] and Zodiac [137], together with another two algorithms that infer multiple aspects in metadata (type, location, equipment, etc)—ProgSyn [136] and Scrabble [138]. Although the latter two are designed to learn all aspects of metadata, we make each to infer only the point type in this set of experiments. We run each algorithm on four different buildings, starting with zero training examples, and calculate the MicroF1 and MacroF1 of inferred type labels. The results are shown in Fig. 7.4a.

We see that AL_Hong marks a stark contrast to all the other algorithms for its steep learning rate (for MicroF1) in the early stage for the first 75 examples. This is because of its clustering-based example selection strategy, which excels in quickly selecting examples that represent a large population and are also informative for model training. However, we do also see that Zodiac and Scrabble are able to catch up after 75 to 125 examples, surpassing in MacroF1, and even achieve a 100% F1 in some case (on building A-1) after converging. These results suggest that Zodiac and Scrabble are better in learning the minor point type classes that appear less frequently in a building, which AL_Hong is not able to learn even with more examples. We would also like to point out the fact that, due to the deterministic nature of the algorithm, ProgSyn and Zodiac may terminate early (e.g, for C-1). Zodiac runs with a preset confidence threshold, and as it gradually acquires training examples, whenever the algorithm has high enough confidence in every testing instance, it will cease. For ProgSyn, it decides whether the learned rules are able to parse every example and stops when it becomes the case. Furthermore, there is no clear winner in this set of experiments. The implication here, however, is that if one wants to quickly label the types with reasonably high accuracy (e.g., around 85%), AL_Hong is an appropriate choice. When one desires better coverage of less frequent types in a long run, Zodiac or Scrabble would be a better choice.

Jump-started Active Learning

All the original active learning based algorithms [137, 135, 138] are designed to work only within the same building, meaning that they do not consider or leverage any information from existing buildings. However, because the inferencerdesign in Plastermakes it convenient to start from any training set, we will next show how the inference results change if we run an active learning based algorithm using information transferred from another building for inferencerinitialization. More specifically, we use another building's point names along with the labels (e.g., from A-1) to formulate the initial training set for an inferencerand after that run the algorithm on another building (e.g., C-1) as we did in Section 7.3.2. The results are shown in Fig. 7.4b.

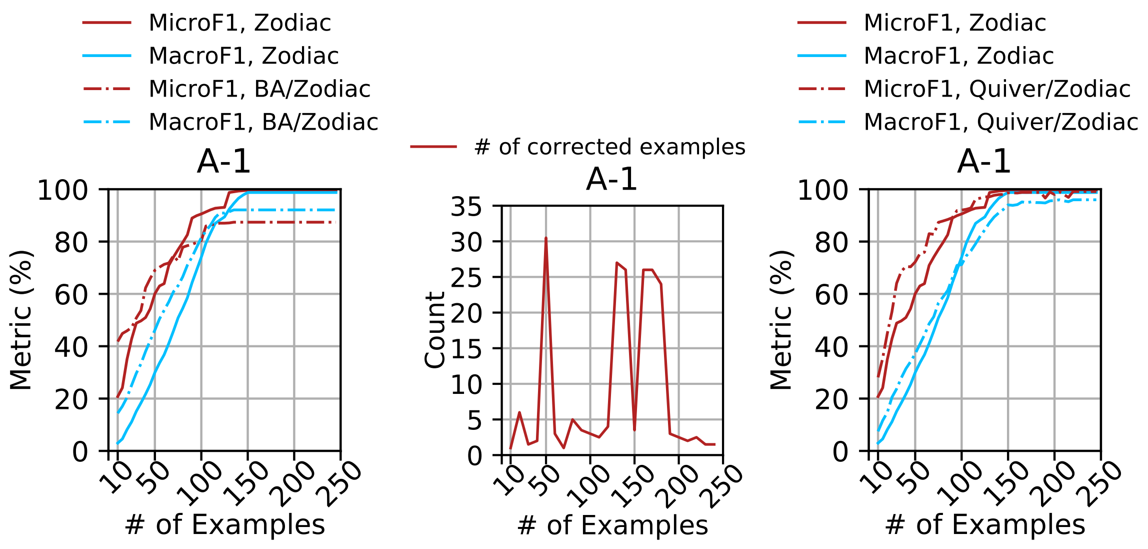
When added with a building from a different vendor with almost completely distinct naming conventions (e.g., A-1 $\rightarrow$ C-1 and C-1 $\rightarrow$ A-1), the type inference performance either remains unchanged or even deteriorates in the early stage. This is expected as adding another building using a different metadata naming convention would introduce more irrelevant patterns to the same point type for the algorithm to learn, which is almost equivalent to injecting noise. Nonetheless, we still notice an increase in MicroF1 for Scrabble in the early stage in the case of A-1 $\rightarrow$ C-1. This is largely due to Scrabble's underlying deep network based representation learning that is able to learn more general patterns even given different buildings. On the other hand, when we add a building from the same vendor which uses a similar vocabulary for point types (see A-2 $\rightarrow$ A-1), we observe a better starting point (71% in the first figure in 7.4b vs 58% in the first figure in 7.4a) and also a better converged MicroF1 for AL_Hong. We observe similar improvements for Scrabble and Zodiac in this case. These results suggest that having labeled buildings with similar naming convention is useful for normalizing subsequent buildings by transferring the information in the raw metadata.

Active Learning for Multiple Entities

Even though detecting the point type is important, other types of entities encoded in the metadata, such as the associated room and equipment, are also essential for building applications. We therefore evaluate Scrabble [138] and ProgSyn [136] for their ability to identify multiple types of entities from the given raw metadata, including the point type, room location, and associated equipment. As shown in Fig. 7.4c, Scrabble outperforms ProgSyn in both MacroF1 and example-level accuracy. The gain in performance of Scrabble is attributed to its more sophisticated representation learning procedure where it first maps input to an intermediate representation and then to actual labels, while ProgSyn maps the raw metadata directly to final labels. For example, for a string `ZNT`, Scrabble first learns its nuanced character-level BIO tags and then maps to the Brick tagset (i.e.e, `Zone Temperature Sensor`), while ProgSyn directly learns its mapping rule to the tagset via regular expressions.

7.3.3 Workflow

Having seen the results on comparing different algorithms individually, we next show how they can interact with each other in Plaster. A key feature of Plasteris the ability to try out different workflows, or pipelines, which integrate different algorithms in different orders. We now present and evaluate three exemplary workflows where two inferencers are connected together for better performance than if they were used independently.



(a) BA → Zodiac (b) Zodiac → Scrabble (c) Quiver → Zodiac

Figure 7.5: Learning efficiency for inferring point type by different workflows. They all show the synergistic improvement in performance. (BA stands for Building Adapter.)

Transfer Learning + Active Learning

A completely automated method such as Building Adapter (BA) is able to achieve relatively high inference precision in point types on a target building, though only for a fraction of the points. It would be natural to connect and feed the labeled examples by BA to an active learning-based method, such as Zodiac [137], as a better starting point. This appears to be similar to the jump-started active learning scenario in Section 7.3.2, in that both provide a better starting point for active learning. However, a fundamental difference here lies in the fact that a method such as BA, which transfers the learnt model via *timeseries data* from a different building to facilitate another learning process based on *textual data*, which is independent from these two buildings' naming conventions, while in the previous scenario we will only see benefits when transferring from a building with a similar naming convention. We implement such a workflow of combining BA and Zodiac to again infer point types, with Fig. 7.5a showing the comparison results. We see that the combination achieves both higher MicroF1 and MacroF1 up to 70 examples, benefiting from the transferred information. However, the incorrect labels from BA's predictions (though only a handful) remain as negative training examples to Zodiac and it cannot recover from such noise in such a naïve integration. These inherited incorrect labels would be corrected or filtered out at the beginning if Zodiac could have the ability to quantify BA's results based on its own criterion. Yet, this will require additional modifications to the original algorithm and is hence out of the scope of Plaster.

High Precision + High Recall

Some algorithms have high precisions while others have high recalls in their inference results. For example, Zodiac [137] has a high precision for point type inference while Scrabble [138] can identify multiple kinds of entities likely with a high recall. Thus we can filter Scrabble's results by using Zodiac's results without compromising the results of either. More specifically, we feed Zodiac's results to Scrabble's prediction and if there is a disparity between the two on an instance, Scrabble will adopts Zodiac's prediction for point type. As shown in Fig. 7.5b, we see there are about 1,500 corrections made to the point type predictions in total (note that we only count the number of corrections made by this strategy, and an instance could be corrected multiple times) with little additional computational cost.

Active Learning + Relationship Learning

Learning functional relationships often relies on perturbations to the control systems (e.g., Quiver [139]) and, to correctly perform perturbations on a target point such as a VAV on/off command, it requires knowing the point types apriori. Thus, it is natural to apply an active learning algorithm (Zodiac) to infer point types as a prior step to a perturbation-based relationship inference algorithm (Quiver). Furthermore, the inferred relationships can in return help examine whether the point types are correctly inferred by an active learning method. For example, the *fact* that a VAV typically contains only one for each type of its sensing and control points can help identify mistakes made in type inference. For example, based on the manual perturbation to a VAV on/off command, Quiver [139] identifies a group of co-located points and finds that there are two supply air temperature sensors; it is highly likely that Zodiac has made a mistake in the type inference. For this experiment, we emulate the above procedure by first running Zodiac to infer point types, and for each predicted VAV on/off command, we use the ground-truth for the co-located points in that VAV (since we are not able to actually run Quiver) and examine if there is any duplicated type among these points, in order to correct any type mis-predictions. We see modest improvement in the type inference results because we only consider $\sim$15 most common point types existing in VAVs, as Quiver can only find the co-located points for VAVs. Although a workflow as such exploits certain domain knowledge, we believe that it will be generally useful to practitioners with special demands in building applications.

7.3.4 Timeseries Feature Selection

We empirically inspect how well each timeseries feature set performs and how effective the feature selection is in Plaster. To this end, we create a workflow that feeds the timeseries data to each of the feature extraction modules included in Plaster, passes the features to a random forest classifier (which is identified as the best

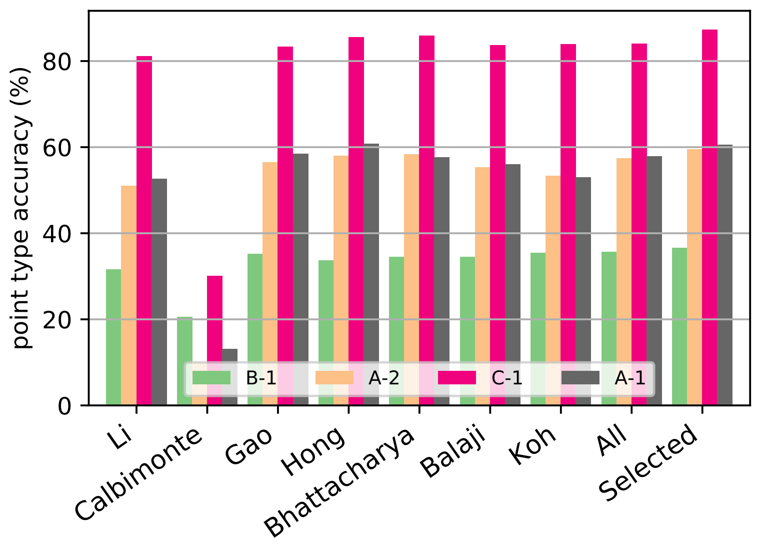


Figure 7.6: Results for timeseries data-based type inference: We show 7 different feature sets (each is shown as a group of bars and each bar represents the result on a building), along with a fusion of them (All) and a better subset via automated feature selection (Selected).

performing classifier in prior work [156]), and predicts the point type for evaluation. We apply the workflow to each of the buildings with timeseries data available and conduct 10-fold cross validation to obtain the point type prediction accuracy. Fig. 7.6 summarizes our results. We observe that each individual feature set roughly performs on a par except the second set. A simple fusion of all the dimensions from each feature set (marked as All in the figure), which equates to a 106-dimensional feature set, does not yield much better performance. However, the selected set of features does give a 3% increase overall than the best set with the number of features reduced to 60. This demonstrates the effectiveness of feature selection and integration provided by Plaster.

Using timeseries feature for point type inference on each building, we clearly notice the performance is far less competitive than the one we obtained by using textual metadata (as demonstrated in Fig. 7.4). However, the implication is that, as data features better suit transfer learning based tasks (according to Table 3.3), the better feature set we have identified here would help to better improve such a procedure, for instance Building Adapter. Moreover, we would also like to emphasize that subsequent users and/or researchers can easily register their own feature set in the feature extractor interface in Plaster, and can also perform feature selection with our provided method to obtain a even better set of features for their target metadata inference problem.

7.4 Discussion and Future Work

First, although not being the focus of Plaster, we see that for the same metadata inference algorithm, still its performance varies significantly from building to building. When designing an algorithm in the future, one would want to test on a diverse set of buildings, rather than one or two, to avoid over-fitting for a single building. Secondly, having seen the synergy between different algorithms via our workflow design, the original authors or subsequent researchers, might consider revisiting and refining the design of existing metadata inference algorithms, in order to better leverage the advantages from each other. Thirdly, given the recent advances in deep neural networks concerning both textual and timeseries data, it remains open how to best harvest their progress to complement or reshape the research on metadata inference. Plasternow provides some basic functions for feature selection taking advantage of existing neural network algorithms. And the architecture of Plasteris flexible enough to directly develop neural network based metadata inference algorithms. Additionally, as Plastercurrently is only compatible with inputs in the Brick format, in future we would want to extend it to exploit existing resources available in other schemata such as Haystack, in order to augment the existing methods in Plastervia a workflow. Furthermore, we would like to provide the ability of automatically selecting an inferencer, or composing a workflow, for a user based on their demands and what are available to them, i.e., the idea of meta-learning [164]. For example, if they want to convert a building with a few others available already, we could connect Building Adapter and Scrabble as a workflow for them.

7.5 Summary

In face of the metadata inference challenge, various methodologies have been proposed to (partially) automate the process of generating structured metadata. Yet, the lack of compatibility between methods precludes the possibility of combining or even comparing them, due to the heterogeneity in the inference scope, input/output format and structure, algorithm interface, evaluation metric. In this chapter, we present the design, implementation, and evaluation of Plaster — a unified, extensible and modular framework that incorporates existing advances in metadata inference. The modular design of Plaster enables generic, flexible workflows as well as development and evaluation of new algorithms. Via systematic evaluations with a set of unified metrics as well as building datasets, we have demonstrated that the workflows in Plaster can, *for the first time ever*, provide a standard benchmark for comparing different metadata inference methods, and can leverage and integrate existing methods in a complementary manner to deliver superior results. We believe Plaster provides a comprehensive framework for further development of new algorithms, techniques, and workflows for metadata inference, as well as for mapping them to a structured ontology like Brick, enabling seamless smart buildings applications in the future.

Chapter 8

Conclusion

8.1 Summary and Key Contributions

In this dissertation, we present the design, implementation, and evaluation of *Data Holmes* — a framework that infers the contextual information about sensors in commercial buildings, requiring minimal to no human input. This is a first step towards the of vision of scalable and fast building analytics. As part of the framework, we describe the four algorithms, together with an open-sourced programming platform, for practitioners such as building managers and researchers to tackle the building metadata problem and foster new algorithms. In summary, we make a few contributions in this dissertation.

8.1.1 A Fully Automated Type Classification Technique

Building Adapter automatically infers the type information in a building *without* requiring human labeling input, leveraging the information from one already mapped building. We provide experimental evidence that the technique is able to automatically label more than 36% percent of the points in a new building with at least 85% accuracy, and for some cases up to 81% points with more than 96% accuracy. We also illustrate how combining multiple buildings as the learning source could further boost the performance (i.e., create more accurate labels for larger fraction of the points), which demonstrates promise in the technique. Furthermore, the technique is general, simple yet effective, which we believe can act as a tool for type inference for not only building sensors, but also in the broader context of the Internet of Things.

8.1.2 A Semi-Automated Type Classification Technique

Due to the fact that no two buildings are completely identical, the first technique cannot create type labels for all the points given a building; labeling the remaining unlabeled points would still require human input, which is often prohibitive. To address this need, we introduce a novel, effective yet general active learning method to predict the sensor types by exploiting the clustering structure in the sensor names, requiring the minimal amount of human input . Extensive experimental comparisons demonstrate the superiority of the technique — it is able to achieve more than 92% accuracy for type classification with at least 20% fewer labeled examples than baselines; on average, we need to present only $\sim$20 examples from a human expert for labeling, to achieve 90% prediction accuracy. In addition, we also demonstrate that how our solution can complement the fully automated technique. Our proposed active learning algorithm is general, and therefore it is applicable in a broader contexts such as document categorization and image retrieval.

8.1.3 A Cross-Predictive Clustering Technique

The proliferation of cheap, ubiquitous sensing infrastructure has permeated the world, and the data in this paradigm is expected to be high-dimensional — the number of (sensor) time series is much larger than the length of time series. Faced with the challenge, we describe a clustering method built upon a new similarity measure, which quantifies the "cross-predictability" between time series, i.e., the degree to which a future value in each time series is predicted by past values of the others. We also provide a theoretical proof that the proposed algorithm will successfully recover the clustering structure in the data with high probability under certain conditions. Experiments on both synthetic and real-world data verify the correctness of our findings, and demonstrate the effectiveness of the algorithm. For the real-world task of sensor type clustering, our method is able to outperform the state-of-art baselines by more than 20% with regard to the quality of identified type clusters.

8.1.4 An Automatic Solution to Functional Relationship Inference

In addition to the type information of sensors, the functional relationships among equipment (e.g., the connection between AHUs and VAVs) is also critical to lots of building applications. We propose an automatic solution to this problem — it first develops a Markovian Event Model to identify the latent events in the time series data of equipment, and then uses a correlation-based metric to locate the correlated set of events for identifying the underlying relations. We demonstrate that the solution achieves an average accuracy 94.38%, compared to the 85.49% by the best baseline. We believe the solution is promising and can be generally applied to event detection and relation inference in broader domains.

8.1.5 An Integration Framework for Metadata Inference Algorithms

Various methods, together with the ones in this dissertation, have been proposed to (partially) automate the process of generating structured metadata. However, the lack of compatibility between methods precludes the possibility of combining or even comparing them, due to their heterogeneity. We design and implement Plaster — a unified, extensible and modular framework that incorporates existing advances in metadata inference. The modular design of Plaster enables generic, flexible workflows as well as development and evaluation of new algorithms. We demonstrate that the workflows in Plaster can, for the first time, provide a standard benchmark for comparing different metadata inference methods, and can leverage and integrate existing methods in a complementary manner to deliver superior results. We believe Plaster can spur and facilitate further development of new algorithms for metadata inference, which would enable seamless smart buildings applications in the future.

8.2 Future Improvements

In this section, we discuss future work that further improves the proposed techniques in this dissertation.

First, we need better representation of the data. The performance of the transfer learning- and active learning-based classification methods is fundamentally bounded by the base classifiers, which rely on a set of general, standard statistical features. The domain specific line of work on representing time series with discretized symbols (e.g., the SAX [165]) or primitive "shapes" (e.g., the "shapelets" [166, 167]) could be revisited and further extended for the sensor time series concerned in this thesis. Recent advances in learning representation of data using deep networks [168, 169, 170, 171] may also contribute to better base classifiers for our method.

Second, we could further reduce the human labeling effort for the semi-automated technique. Currently, when acquiring labels for the selected examples, the active learning method only employs manual human input. However, provided that we often have already well trained classifiers from other buildings (as in the Building Adapter), we could use their outputs as the labels for active learning, instead of always resorting to a human labeler. Yet, we would need to accurately quantify the credibility of these transferred "labelers".

Third, we could extend the cross-predictive clustering method to more complicated settings. The current setting of the method assumes no noise or outliers, which often does not hold in practice. We could extend the assumption to allow noise or outliers in the data distribution, yet still bearing the theoretical guarantee; and this is a fairly understudied direction, under our VAR data model assumption.

Fourth, the event detection model needs further development and comprehensive evaluation. The model currently considers each sensor time series independently to detect events, and it would be beneficial to incorporate the dependency between sequences into the modeling, such as VAVs connected to the same AHU might respond to the same set of events. In addition, we would also like to evaluate the method on data containing events in broader scenarios, e.g., events in water usage traces or events in smart wearable records, in order to demonstrate general usability of the method.

8.3 Future Research Directions

We also discuss a few future research directions inspired by the work in this dissertation.

The main focus of the techniques proposed in this dissertation is to infer the contextual information about sensors in commercial buildings. However, the concept of "data context" definitely applies to beyond just building data, and we expect our techniques to be generally applicable, too. As each individual starts to own a fleet of personal devices that produce a plethora of digital traces, being able to automatically associate the data with the owner and to interpret the data will be a key to future data intelligence. For example, clinical records for each patient are still mostly manually created and maintained, while the sensory measurements of the same patient are often (automatically) digitally stored; matching the two sources is a first step to patient identification and is currently still manually performed. With the advent of Internet of Things (IoT), we would expect increasing demand and more stringent requirement of such data context generation and analysis, yet requiring the minimal amount of human input.

Another direction is concerned with the security and privacy in the process of exploiting data. Data-driven analysis has proven to be effective in revealing a lot about the data sources. For example, BACnet in buildings is known to be hackable and by simply inspecting the occupancy sensor trace in an office room, someone could easily infer the occupant's daily schedule and maliciously pay a visit to the occupant's home while he is at work. Therefore, how to still fulfill computational requirements such as extracting effective features without revealing data owner's privacy has garnered attentions. Techniques including differential privacy have been put into extensive use; however, how to effectively protect and secure users' data in the context of IoT, given the possible exponentially long chain of data stakeholders from data collectors to storage service to analytics providers, remains a challenge.

Appendix A

Proof of Theorem 5.2.3

To prove the theorem, we need to prove the two conditions are satisfied as required in definition 5.2.1 of cluster recovery property (CRP), i.e., the optimal solution is *nonzero* and has support indices of columns only *from the same cluster* (SRP - the self-reconstruction property). At the core of our proof lies the primal-dual witness method [172] widely used in lasso analysis, which involves the explicit construction of an optimal primal-dual problem and a dual certificate. We organize the proof as follows:

(1) We prove the *SRP* by duality: first we establish a set of conditions on the optimal dual variable, which corresponds to the case where all the primal solutions satisfy SRP; we then construct a dual variable to certify the proof. This step includes §A.1, §A.2 and §A.3.

(2) We prove the optimal solution is *nonzero* by showing that there exists a value smaller than the value of (A.2.1) when the solution is zero. This step corresponds to §A.4.

(3) We derive the range of λ for successfully recovering the clustering structure: the proof of step (1) imposes an upper bound on λ while step (2) reduces to a lower bound of λ. It is essential to require the lower bound to be smaller than the upper bound, and based on this we will derive a sufficient condition that guarantees the existence of a valid λ. This step will be explained in §A.5.

A.1 Optimal Primal-Dual Problem

We propose a regularized Dantzig selector and define our primal convex optimization problem as the following general form:

$$\mathbf{P}_0 : \arg\min_{\boldsymbol{\beta}_i} \lambda \|\hat{\boldsymbol{\Sigma}}\boldsymbol{\beta}_i - \hat{\boldsymbol{\gamma}}_i\|_{\infty,\infty} + \|\boldsymbol{\beta}_i\|_1, \tag{A.1.1}$$

where $\hat{\boldsymbol{\Sigma}} = \mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}}/(T-1)$ and $\hat{\gamma}_i = \mathbf{X}_{\mathcal{S}}^\top (\mathbf{X}_{\mathcal{T}})_{*i}/(T-1)$. Then the dual problem of (A.1.1) is:

$$\mathbf{D}_0 : \arg\max_{\boldsymbol{\nu}} \ \langle \boldsymbol{\nu}, \hat{\gamma}_i \rangle \quad \text{subject to} \quad \|\boldsymbol{\nu}\|_1 = \lambda, \ \|\hat{\boldsymbol{\Sigma}}\boldsymbol{\nu}\|_{\infty,\infty} \le 1.$$

To avoid cluttered notations, we omit some subscriptions and use $\hat{\gamma}$ to refer to $\hat{\gamma}_i$ unless stated otherwise, and likewise use $\hat{\gamma}_{\mathcal{S}_l}$ for $(\hat{\gamma}_{\mathcal{S}_l})_i$ in the rest of the proof. We state the following technical lemma on the support of the optimal solution, which extends the Lemma 7.1 in [117]:

Lemma A.1.1. *Consider a vector $\boldsymbol{\gamma} \in \mathbb{R}^d$ and a matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$. If there exists a pair $(\boldsymbol{\beta}, \boldsymbol{\nu})$ with $\boldsymbol{\beta}$'s support $S \subseteq T$, and a dual certificate vector $\boldsymbol{\nu}$ satisfying:*

$$\boldsymbol{\Sigma}_{S,\eta}\boldsymbol{\nu}_\eta + sgn(\boldsymbol{\beta}_S) = 0, \quad \|\boldsymbol{\nu}\|_1 = \lambda, \quad \|\boldsymbol{\Sigma}_{T\cap S^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty} \le 1, \quad \|\boldsymbol{\Sigma}_{T^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty} < 1,$$

where η is the set of indices of entry i such that $|(\boldsymbol{\Sigma}\boldsymbol{\beta} - \boldsymbol{\gamma})_i| = \|\boldsymbol{\Sigma}\boldsymbol{\beta} - \boldsymbol{\gamma}\|_{\infty,\infty}$, then all the optimal solutions $\boldsymbol{\beta}^*$ to (A.1.1) *obey $\boldsymbol{\beta}^*_{T^c} = 0$.*

The detailed proof of this lemma is deferred to Section B.

A.2 Construction of the Dual Certificate

We define $\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l} = (\mathbf{X}_{\mathcal{S}}^l)^\top \mathbf{X}_{\mathcal{S}}^l \in \mathbb{R}^{d_l\times d_l}$ and $\hat{\gamma}_{\mathcal{S}_l} = (\mathbf{X}_{\mathcal{S}}^l)^\top (\mathbf{X}_{\mathcal{T}}^l)_{*i} \in \mathbb{R}^{d_l}$, and we next apply the above lemma to $\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}$ and $\hat{\gamma}_{\mathcal{S}_l}$ for a particular cluster $\mathcal{C}_l$, and prove the existence of a pair $(\hat{\boldsymbol{\beta}}, \boldsymbol{\nu})$ such that the optimal solution $\hat{\boldsymbol{\beta}}$ satisfies the self-reconstruction property and the certificate $\boldsymbol{\nu}$ satisfies all the conditions in Lemma A.1.1. Thereafter, the SRP holds for any optimal solution to (A.1.1).

To construct the dual certificate, we consider the following problem:

$$\mathbf{P}_1 : \ \hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \ \lambda\|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}\boldsymbol{\beta} - \hat{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} + \|\boldsymbol{\beta}\|_1, \tag{A.2.1}$$

and its dual form is:

$$\mathbf{D}_1 : \arg\max_{\boldsymbol{\nu}} \ \langle \boldsymbol{\nu}, \hat{\gamma}_{\mathcal{S}_l} \rangle \quad \text{subject to} \quad \|\boldsymbol{\nu}\|_1 = \lambda, \ \|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}\boldsymbol{\nu}\|_{\infty,\infty} \le 1.$$

Define $\hat{\boldsymbol{\beta}}$ to be the optimal solution to the primal problem $\mathbf{P}_1$, the dual problem $\mathbf{D}_1$ is then also feasible by strong duality, which implies that for each optimal solution $\hat{\boldsymbol{\beta}}$ to (A.2.1) with its support on $\hat{S}$, there exists a dual certificate $\hat{\boldsymbol{\nu}}$ that by definition satisfies:

$$\hat{\boldsymbol{\Sigma}}_{\hat{S},\hat{\eta}}\hat{\boldsymbol{\nu}}_{\hat{\eta}} + sgn(\hat{\boldsymbol{\beta}}_{\hat{S}}) = 0, \quad \|\hat{\boldsymbol{\nu}}\|_1 = \lambda, \quad \|\hat{\boldsymbol{\Sigma}}_{\hat{S}^c,\hat{\eta}}\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty} \leq 1, \tag{A.2.2}$$

where $\hat{\eta}$ is the set of indices of entry i such that $\left|(\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\gamma}}_{\mathcal{S}_l})_i\right| = \|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\gamma}}_{\mathcal{S}_l}\|_{\infty,\infty}$. If we define:

$$\boldsymbol{\beta} = \left[\mathbf{0}, ..., \mathbf{0}, \hat{\boldsymbol{\beta}}, \mathbf{0}, ..., \mathbf{0}\right] \text{ and } \boldsymbol{\nu} = \left[\mathbf{0}, ..., \mathbf{0}, \hat{\boldsymbol{\nu}}, \mathbf{0}, ..., \mathbf{0}\right]. \tag{A.2.3}$$

Note that if we use $\hat{T}$ to index the columns of $\mathbf{X}_{\mathcal{S}}$ from cluster $\mathcal{C}_l$, it is easy to see that the $\boldsymbol{\beta}$ defined above obeys $(\boldsymbol{\beta})_{\hat{T}^c} = \mathbf{0}$ (i.e., the SRP holds), and we have $\hat{T} \cap \hat{S}^c = \hat{S}^c$, which makes the last condition in (A.2.2) equivalent to the third condition as required in Lemma A.1.1. Therefore, it remains to check the following condition according to the lemma:

$$\|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l^c,\hat{\eta}}\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty} < 1. \tag{A.2.4}$$

If the condition is satisfied, we can then conclude that $\boldsymbol{\nu}$ as constructed in (A.2.3) is a dual certificate as required in Lemma A.1.1, and the SRP holds for all the optimal solutions.

A.3 Upper Bound of λ

In this section, we show that (A.2.4) can be separated into two terms and bounded individually, we then further impose inequality on the bound to obtain the desired condition, which boils down to an upper bound of λ.

Following (A.2.4), we have:

$$\begin{aligned}\|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l^c,\hat{\eta}}\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty} &= \|(\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l^c,\hat{\eta}} - \boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}} + \boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}})\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty} \\ &\leq \|(\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l^c,\hat{\eta}} - \boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}})\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty} + \|\boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty}, \end{aligned} \tag{A.3.1}$$

where we applied the triangle inequality for the last inequality. To bound the first term, we introduce the following lemma from [113]:

Lemma A.3.1. *Let $\hat{\boldsymbol{\Sigma}}$ be the sample covariance matrix and $\boldsymbol{\Sigma}$ be the population covariance matrix, when $T \geq max(6\log d, 1)$ for the sample data matrix $\mathbf{X} \in \mathbb{R}^{T\times d}$, we have the following condition hold with probability at least $1 - 6d^{-1}$:*

$$\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\|_{\infty,\infty} \leq \frac{16\|\boldsymbol{\Sigma}\|_2 \max_j(\Sigma_{jj})}{\min_j(\Sigma_{jj})(1 - \|\mathbf{A}\|_2)}\sqrt{\frac{6\log d + 4}{T}}. \tag{A.3.2}$$

Particularly, for the ease of reference, we write the RHS of (A.3.2) as ρ. Therefore, we have:

$$\|(\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l^c,\hat{\eta}} - \boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}})\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty} \leq \|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l^c,\hat{\eta}} - \boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty} \cdot \|\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_1 \leq \rho\|\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_1 \leq \rho\|\hat{\boldsymbol{\nu}}\|_1 \leq \rho\lambda, \tag{A.3.3}$$

where we applied Lemma A.3.1 for the second inequality and the second condition in (A.2.2) for the last inequality. Next we consider the second term in (A.3.1):

$$\|\boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_{\infty,\infty} \leq \|\boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty} \cdot \|\hat{\boldsymbol{\nu}}_{\hat{\eta}}\|_1 \leq \lambda\|\boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty}, \tag{A.3.4}$$

where the last inequality again follows from the second condition in (A.2.2). Putting together (A.3.1), (A.3.3) and (A.3.4), we get a sufficient condition for (A.2.4):

$$\rho\lambda + \lambda\|\boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty} < 1,$$

which implies:

$$\lambda < \frac{1}{\rho + \|\boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty}}.$$

To generalize the above condition to all the clusters $l = 1, 2, \ldots, k$, we further need:

$$\lambda < \min_l \frac{1}{\rho + \|\boldsymbol{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty}}. \tag{A.3.5}$$

A.4 Lower Bound of λ

In this section, we set to prove the nontrivialness of the optimal solution to (A.2.1), i.e., the solution is nonzero. We will show that this step reduces to a lower bound of λ, and we state that if λ satisfies the following condition, the optimal solution to (A.2.1) can not be trivial, i.e., $\boldsymbol{\beta} = 0$ can never be optimal:

$$\lambda > \frac{1}{r(\mathcal{P}((\boldsymbol{\Sigma}_0)_{\mathcal{S}_l,\mathcal{S}_l}))r(\mathcal{P}((\boldsymbol{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}))(\|\boldsymbol{\gamma}\|_{\infty,\infty} - 2\rho) - \rho}.$$

On the one hand, if $\boldsymbol{\beta} = \mathbf{0}$ is the optimal solution to (A.2.1), then the optimal value will be $\lambda\|\hat{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty}$; on the other hand, suppose we can choose a particular $\boldsymbol{\beta}^*$ such that:

$$\lambda\|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}\boldsymbol{\beta}^* - \hat{\boldsymbol{\gamma}}_{\mathcal{S}_l}\|_{\infty,\infty} + \|\boldsymbol{\beta}^*\|_1 < \lambda\|\hat{\boldsymbol{\gamma}}_{\mathcal{S}_l}\|_{\infty,\infty}. \tag{A.4.1}$$

If we can prove the existence of such a $\boldsymbol{\beta}^*$, we are then equivalently proving $\boldsymbol{\beta} = \mathbf{0}$ is not optimal. In particular, we let $\boldsymbol{\beta}^*$ be the optimal solution to the following fictitious problem:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 \quad \text{subject to} \quad \boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}\boldsymbol{\beta} = \boldsymbol{\gamma}_{\mathcal{S}_l}, \tag{A.4.2}$$

and its dual is:

$$\arg\max_{\boldsymbol{\nu}} \langle \boldsymbol{\nu}, \boldsymbol{\gamma}_{\mathcal{S}_l} \rangle \quad \text{subject to} \quad \|\boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}\boldsymbol{\nu}\|_{\infty,\infty} \le 1. \tag{A.4.3}$$

Here $\boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l} \in \mathbb{R}^{d_l \times d_l}$ and $\boldsymbol{\gamma}_{\mathcal{S}_l} \in \mathbb{R}^{d_l}$ is a column of $\boldsymbol{\Sigma}_1$. Based on the condition $\boldsymbol{\Sigma}\mathbf{A}^\top = \boldsymbol{\Sigma}_1$, it is valid to assume there exists such a $\boldsymbol{\beta}^*$.

Next, we first deal with the LHS of (A.4.1):

$$\begin{aligned}
&\lambda\|\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}\boldsymbol{\beta}^* - \hat{\boldsymbol{\gamma}}_{\mathcal{S}_l}\|_{\infty,\infty} + \|\boldsymbol{\beta}^*\|_1 \\
&= \|\boldsymbol{\beta}^*\|_1 + \lambda\|(\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l} - \boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l} + \boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l})\boldsymbol{\beta}^* - (\hat{\boldsymbol{\gamma}}_{\mathcal{S}_l} - \boldsymbol{\gamma}_{\mathcal{S}_l} + \boldsymbol{\gamma}_{\mathcal{S}_l})\|_{\infty,\infty} \\
&= \|\boldsymbol{\beta}^*\|_1 + \lambda\|(\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l} - \boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l})\boldsymbol{\beta}^* - (\hat{\boldsymbol{\gamma}}_{\mathcal{S}_l} - \boldsymbol{\gamma}_{\mathcal{S}_l})\|_{\infty,\infty} \\
&\le \|\boldsymbol{\beta}^*\|_1 + \lambda\|(\hat{\boldsymbol{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l} - \boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l})\boldsymbol{\beta}^*\|_{\infty,\infty} + \lambda\|(\hat{\boldsymbol{\gamma}}_{\mathcal{S}_l} - \boldsymbol{\gamma}_{\mathcal{S}_l})\|_{\infty,\infty} \\
&\le \|\boldsymbol{\beta}^*\|_1 + \lambda\rho\|\boldsymbol{\beta}^*\|_1 + \lambda\rho \\
&= \|\boldsymbol{\beta}^*\|_1(1+\rho\lambda) + \rho\lambda,
\end{aligned} \tag{A.4.4}$$

where the first inequality follows again from the triangle inequality while the second inequality is based on the bound as given in Lemma A.3.1. To obtain an upper bound of $\boldsymbol{\beta}^*$, we rely on the use of polar set, inradius and circumradius as defined in Section B.1. Based on the condition in (A.4.3) and according to definition B.1.1, we have:

$$\|\boldsymbol{\nu}\|_2 \le R([\mathcal{P}(\boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l})]^\circ) = \frac{1}{r(\mathcal{P}(\boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}))}, \tag{A.4.5}$$

where we applied Lemma B.1.3 for the equality. Note that, $\boldsymbol{\gamma}_{\mathcal{S}_l}$ is a column of $(\boldsymbol{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}$ and according to the definition of circumradius, we have:

$$\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_2 \le R([\mathcal{P}((\boldsymbol{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l})]) = R([\mathcal{P}((\boldsymbol{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l})]^\circ) = \frac{1}{r(\mathcal{P}((\boldsymbol{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}))}. \tag{A.4.6}$$

The first equality holds because $\mathcal{P}(\boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l})$ is convex itself in the first place, and again we applied Lemma B.1.3 for the last equality. Since (A.4.2) is a linear program, strong duality holds:

$$\|\boldsymbol{\beta}^*\|_1 = \langle \boldsymbol{\nu}, \boldsymbol{\gamma}_{\mathcal{S}_l} \rangle \le \|\boldsymbol{\nu}\|_2 \cdot \|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_2 \le \frac{1}{r(\mathcal{P}(\boldsymbol{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}))r(\mathcal{P}((\boldsymbol{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}))}. \tag{A.4.7}$$

For the last inequality, we applied (A.4.5) and (A.4.6). Plugging (A.4.7) into (A.4.4), we get:

$$\lambda\|\hat{\mathbf{\Sigma}}_{\mathcal{S}_l,\mathcal{S}_l}\boldsymbol{\beta}^* - \hat{\boldsymbol{\gamma}}_{\mathcal{S}_l}\|_{\infty,\infty} + \|\boldsymbol{\beta}^*\|_1 \leq \frac{1+\rho\lambda}{r(\mathcal{P}(\mathbf{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}))r(\mathcal{P}((\mathbf{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}))} + \rho\lambda. \tag{A.4.8}$$

Next, we consider the RHS of (A.4.1):

$$\begin{aligned}\lambda\|\hat{\boldsymbol{\gamma}}_{\mathcal{S}_l}\|_{\infty,\infty} &= \lambda\|\boldsymbol{\gamma}_{\mathcal{S}_l} + \hat{\boldsymbol{\gamma}}_{\mathcal{S}_l} - \boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} \\ &\geq \lambda\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - \lambda\|\hat{\boldsymbol{\gamma}}_{\mathcal{S}_l} - \boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} \\ &\geq \lambda\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - \rho\lambda, \end{aligned} \tag{A.4.9}$$

where it follows from the triangle inequality and the bound in Lemma A.3.1 for the first and second inequality, respectively. Therefore, to guarantee there exists a $\boldsymbol{\beta}^*$, by putting together (A.4.8) and (A.4.9), a sufficient condition for (A.4.1) to hold is:

$$\frac{1+\rho\lambda}{r(\mathcal{P}(\mathbf{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}))r(\mathcal{P}((\mathbf{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}))} + \rho\lambda < \lambda\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - \rho\lambda,$$

which implies:

$$\lambda > \frac{1}{r(\mathcal{P}(\mathbf{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}))r(\mathcal{P}((\mathbf{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}))(\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho) - \rho}. \tag{A.4.10}$$

A.5 Existence of λ

We compactly denote $\mathcal{P}_0^l = \mathcal{P}(\mathbf{\Sigma}_{\mathcal{S}_l,\mathcal{S}_l}) = \mathcal{P}(\mathbb{E}[\boldsymbol{X}_t^l(\boldsymbol{X}_t^l)^\top])$, $\mathcal{P}_1^l = \mathcal{P}((\mathbf{\Sigma}_1)_{\mathcal{S}_l,\mathcal{S}_l}) = \mathcal{P}(\mathbb{E}[\boldsymbol{X}_t^l(\boldsymbol{X}_{t+1}^l)^\top])$, $r_0^l = r(\mathcal{P}_0^l)$, $r_1^l = r(\mathcal{P}_1^l)$, and $r_0r_1 = \min_l r_0^l r_1^l$. To guarantee there exists a λ satisfying both (A.4.10) and (A.3.5), the following condition needs to hold:

$$\max_l \frac{1}{r_0^l r_1^l(\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho) - \rho} < \lambda < \min_l \frac{1}{\rho + \|\mathbf{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty}}.$$

Also note that:

$$\max_l \frac{1}{r_0^l r_1^l(\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho) - \rho} = \frac{1}{\min_l r_0^l r_1^l(\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho) - \rho}.$$

Therefore, it suffices to require λ obey:

$$\frac{1}{r_0r_1(\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho) - \rho} < \lambda < \frac{1}{\rho + \|\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}\|_{\infty,\infty}}. \tag{A.5.1}$$

Note that we further relaxed $\|\mathbf{\Sigma}_{\mathcal{S}_l^c,\hat{\eta}}\|_{\infty,\infty}$ to $\|\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}\|_{\infty,\infty}$ on the RHS, and (A.5.1) is equivalent to:

$$\rho + \|\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}\|_{\infty,\infty} < r_0 r_1(\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho) - \rho,$$

which implies that:

$$r_0 r_1 > \frac{\|\mathbf{\Sigma}_{\mathcal{S}_l^c,\mathcal{S}_l}\|_{\infty,\infty} + 2\rho}{\|\boldsymbol{\gamma}_{\mathcal{S}_l}\|_{\infty,\infty} - 2\rho}, \tag{A.5.2}$$

with

$$\rho = \frac{16\|\mathbf{\Sigma}\|_2 \max_j \Sigma_{jj}}{\min_j \Sigma_{jj}(1 - \|\mathbf{A}\|_2)} \sqrt{\frac{6\log d + 4}{T}}.$$

Here (A.5.2) is a sufficient condition to guarantee that the range given in (A.5.1) is non-empty. This concludes the proof of Theorem 5.2.3.

Appendix B

Proof of Lemma A.1.1

Proof. Observe that for any optimal solution β^* to problem (A.1.1), we have:

$$\begin{aligned}
&\|\boldsymbol{\beta}^*\|_1 + \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta}^* - \boldsymbol{\gamma}\|_{\infty,\infty} - \|\boldsymbol{\beta}\|_1 - \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta} - \boldsymbol{\gamma}\|_{\infty,\infty} \\
= &\|\boldsymbol{\beta}^*_S\|_1 + \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta}^* - \boldsymbol{\gamma}\|_{\infty,\infty} - \|\boldsymbol{\beta}\|_1 - \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta} - \boldsymbol{\gamma}\|_{\infty,\infty} \\
= &\ \langle sgn(\boldsymbol{\beta}_S), \boldsymbol{\beta}^*_S - \boldsymbol{\beta}_S\rangle + \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta}^* - \boldsymbol{\gamma}\|_{\infty,\infty} - \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta} - \boldsymbol{\gamma}\|_{\infty,\infty} \\
\geq &\ \langle sgn(\boldsymbol{\beta}_S), \boldsymbol{\beta}^*_S - \boldsymbol{\beta}_S\rangle + \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \langle \boldsymbol{\Sigma}_{*,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}^* - \boldsymbol{\beta}\rangle \\
= &\ \langle -\boldsymbol{\Sigma}_{S,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}^*_S - \boldsymbol{\beta}_S\rangle + \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \langle \boldsymbol{\Sigma}_{*,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}^* - \boldsymbol{\beta}\rangle \\
= &\|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \langle \boldsymbol{\Sigma}_{*,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}^*\rangle - \langle \boldsymbol{\Sigma}_{S,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}^*_S\rangle + \langle \boldsymbol{\Sigma}_{S,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}_S\rangle - \langle \boldsymbol{\Sigma}_{*,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}\rangle \\
= &\|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{*,\eta}\boldsymbol{\beta}^*\right\rangle - \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{S,\eta}\boldsymbol{\beta}^*_S\right\rangle + \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{S,\eta}\boldsymbol{\beta}_S\right\rangle - \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{*,\eta}\boldsymbol{\beta}\right\rangle \\
= &\|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T\cap S^c,\eta}\boldsymbol{\beta}^*_{T\cap S^c}\right\rangle + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T^c,\eta}\boldsymbol{\beta}^*_{T^c}\right\rangle .
\end{aligned}$$

The first inequality holds because of the convexity of the original optimization function. The last equality holds because :

$$\boldsymbol{\Sigma}\boldsymbol{\beta}^* = \boldsymbol{\Sigma}^\top_{S,*}\boldsymbol{\beta}^*_S + \boldsymbol{\Sigma}^\top_{T\cap S^c,*}\boldsymbol{\beta}^*_{T\cap S^c} + \boldsymbol{\Sigma}^\top_{T^c,*}\boldsymbol{\beta}^*_{T^c}, \quad \text{and} \quad \boldsymbol{\Sigma}\boldsymbol{\beta} = \boldsymbol{\Sigma}^\top_{S,*}\boldsymbol{\beta}_S.$$

With the third inequality condition stated in Lemma A.1.1, we have:

$$\left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T\cap S^c,\eta}\boldsymbol{\beta}^*_{T\cap S^c}\right\rangle = \langle \boldsymbol{\Sigma}_{T\cap S^c,\eta}\boldsymbol{\nu}_\eta, \boldsymbol{\beta}^*_{T\cap S^c}\rangle \leq \|\boldsymbol{\Sigma}_{T\cap S^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty} \cdot \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 \leq \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1,$$

which implies:

$$\|\boldsymbol{\beta}^*_{T\cap S^c}\|_1(1 - \|\boldsymbol{\Sigma}_{T\cap S^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty}) \geq 0. \tag{B.0.1}$$

In a similar manner, we have $\left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T^c,\eta}\boldsymbol{\beta}^*_{T^c}\right\rangle \leq \|\boldsymbol{\Sigma}_{T^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty} \cdot \|\boldsymbol{\beta}^*_{T^c}\|_1$. We also have $\|\boldsymbol{\beta}_S\|_1 = \|\boldsymbol{\beta}\|_1$ since $\boldsymbol{\beta}$ is supported on S, and therefore we get:

$$\begin{aligned}
&\|\boldsymbol{\beta}^*\|_1 + \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta}^* - \boldsymbol{\gamma}\|_{\infty,\infty} - \|\boldsymbol{\beta}\|_1 - \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta} - \boldsymbol{\gamma}\|_{\infty,\infty} \\
\geq\ & \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 + \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T\cap S^c,\eta}\boldsymbol{\beta}^*_{T\cap S^c}\right\rangle + \|\boldsymbol{\beta}^*_{T^c}\|_1 + \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T^c,\eta}\boldsymbol{\beta}^*_{T^c}\right\rangle \\
\geq\ & \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1 - \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T\cap S^c,\eta}\boldsymbol{\beta}^*_{T\cap S^c}\right\rangle + \|\boldsymbol{\beta}^*_{T^c}\|_1 - \left\langle \boldsymbol{\nu}_\eta, \boldsymbol{\Sigma}^\top_{T^c,\eta}\boldsymbol{\beta}^*_{T^c}\right\rangle \\
=\ & \|\boldsymbol{\beta}^*_{T\cap S^c}\|_1(1 - \|\boldsymbol{\Sigma}_{T\cap S^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty}) + \|\boldsymbol{\beta}^*_{T^c}\|_1(1 - \|\boldsymbol{\Sigma}_{T^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty}) \\
\geq\ & \|\boldsymbol{\beta}^*_{T^c}\|_1(1 - \|\boldsymbol{\Sigma}_{T^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty}),
\end{aligned}$$

where the last inequality follows from the inequality in (B.0.1). Furthermore, based on the last inequality condition stated in Lemma A.1.1, it is easy to see that $(1 - \|\boldsymbol{\Sigma}_{T^c,\eta}\boldsymbol{\nu}_\eta\|_{\infty,\infty})$ is strictly greater than 0. On the other side, using the fact that $\boldsymbol{\beta}^*$ is the optimal solution, $\|\boldsymbol{\beta}^*\|_1 + \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta}^* - \boldsymbol{\gamma}\|_{\infty,\infty} - \|\boldsymbol{\beta}\|_1 - \lambda\|\boldsymbol{\Sigma}\boldsymbol{\beta} - \boldsymbol{\gamma}\|_{\infty,\infty} \leq 0$. Therefore it follows that $\boldsymbol{\beta}^*_{T^c} = 0$. We conclude the proof. □

B.1 Auxiliary Results

We state some necessary definitions and lemmas in this section.

Definition B.1.1 (Polar Set). *The polar set $\mathcal{K}^\circ$ of set $\mathcal{K} \in \mathbb{R}^d$ is defined as:*

$$\mathcal{K}^\circ = \left\{ \boldsymbol{y} \in \mathbb{R}^d \mid \langle \boldsymbol{x}, \boldsymbol{y}\rangle \leq 1 \textit{ for all } \boldsymbol{x} \in \mathcal{K} \right\}.$$

Definition B.1.2 (Circumradius [117]). *The circumradius of a convex body $\mathcal{P}$, denoted by $R(\mathcal{P})$, is defined as the radius of the smallest ball containing $\mathcal{P}$.*

We will also leverage the following lemma:

Lemma B.1.3 ([173], page 448). *For a symmetric convex body $\mathcal{P}$, i.e., $\mathcal{P} = -\mathcal{P}$, the following relationship between the inradius of $\mathcal{P}$ and the circumradius of its polar $\mathcal{P}^\circ$ holds:*

$$r(\mathcal{P})R(\mathcal{P}^\circ) = 1.$$

Bibliography

[1] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. Sentinel: Occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, 2013.

[2] Neil E Klepeis, William C Nelson, Wayne R Ott, John P Robinson, Andy M Tsang, Paul Switzer, Joseph V Behar, Stephen C Hern, and William H Engelmann. The national human activity pattern survey (nhaps): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science and Environmental Epidemiology*, 11(3):231, 2001.

[3] ENERGY STAR program U.S. Environmental Protection Agency. Useful facts and figures. June 2007.

[4] U.S. Department of Energy Better Buildings program. Total annual cost of energy in the commercial and industrial sector. March 2015.

[5] US DOE. Better buildings challenge. *http://www4.eere.energy.gov/challenge/sites/default /files/uploaded-files/may-recognition-fs-052013.pdf (Feb. 26, 2014)*, 2013.

[6] Comfy. `https://gocomfy.com/`.

[7] Bharathan Balaji, Jason Koh, Nadir Weibel, and Yuvraj Agarwal. Genie: a longitudinal study comparing physical and software thermostats in office buildings. In *UbiComp*, pages 1200–1211. ACM, 2016.

[8] Xuesong Liu, Burcu Akinci, Mario Berges, and James H Garrett Jr. An integrated performance analysis framework for hvac systems using heterogeneous data models and building automation systems. In *BuildSys*, pages 145–152. ACM, 2012.

[9] Srinivas Katipamula and Michael R Brambley. Methods for fault detection, diagnostics, and prognostics for building systemsa review, part i. *Hvac&R Research*, 11(1):3–25, 2005.

[10] Commercial Buildings Energy Consumption Survey 2012—Microdata. `https://www.eia.gov/consumption/commercial/data/2012/`, 2012.

[11] Jennifer Warnick. 88 acres: How microsoft quietly built the city of the future. *http://www.microsoft.com/en-us/stories/88acres/88-acres-how-microsoft-quietly-built-the-city-of-the-future-chapter-1.aspx (May 8, 2015)*, 2012.

[12] Anika Schumann, Joern Ploennigs, and Bernard Gorman. Towards automating the deployment of energy saving approaches in buildings. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, 2014.

[13] Marco Pritoni, Arka A Bhattacharya, David Culler, and Mark Modera. Short paper: A method for discovering functional relationships between air handling units and variable-air-volume boxes from sensor data. In *BuildSys*, pages 133–136. ACM, 2015.

[14] Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev, and David Culler. Boss: Building operating system services. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, NSDI'13, 2013.

[15] Andrew Krioukov, Gabe Fierro, Nikita Kitaev, and David Culler. Building application stack (bas). In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '12, 2012.

[16] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *BuildSys*, pages 13–22. ACM, 2015.

[17] Jingkun Gao, Joern Ploennigs, and Mario Berges. A data-driven meta-data inference framework for building automation systems. In *BuildSys*, pages 23–32. ACM, 2015.

[18] Virginia Smith, Tamim Sookoor, and Kamin Whitehouse. Modeling building thermal response to hvac zoning. *ACM SIGBED Review*, 9(3):39–45, 2012.

[19] Dezhi Hong, Jorge Ortiz, Kamin Whitehouse, and David Culler. Towards automatic spatial verification of sensor placement in buildings. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[20] Merthan Koc, Burcu Akinci, and Mario Bergés. Comparison of linear correlation and a statistical dependency measure for inferring spatial relation of temperature sensors in buildings. In *BuildSys*, pages 152–155. ACM, 2014.

[21] Erhard Rahm and Philip A Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[22] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 509–520, New York, NY, USA, 2001. ACM.

[23] Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Halevy, and Pedro Domingos. imap: Discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 383–394, New York, NY, USA, 2004. ACM.

[24] Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Halevy. Corpus-based schema matching. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 57–68, Washington, DC, USA, 2005. IEEE Computer Society.

[25] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22, Oct 2010.

[26] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, New York, NY, USA, 2007.

[27] Rich Caruana. Multitask learning. *Machine Learning*, 28(1), 1997.

[28] Hal Daumé, III and Daniel Marcu. Domain adaptation for statistical classifiers. *J. Artif. Int. Res.*, 26(1), May 2006.

[29] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, 2007.

[30] J. Huang, A.J. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, volume 19. The MIT Press, Cambridge, MA, 2007. Pre-proceedings version.

[31] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2), 2000.

[32] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artif. Intell. Rev.*, 11(1-5), February 1997.

[33] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1), March 1991.

[34] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[35] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1), March 1996.

[36] Yoav Freund, H.Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3), 1997.

[37] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *Proceedings of the 20th Annual Conference on Learning Theory*, COLT'07, Berlin, Heidelberg, 2007. Springer-Verlag.

[38] Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, New York, NY, USA, 2004.

[39] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, New York, NY, USA, 2008.

[40] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[41] Xavier Golay, Spyros Kollias, Gautier Stoll, Dieter Meier, Anton Valavanis, and Peter Boesiger. A new correlation-based fuzzy logic clustering algorithm for fmri. *Magnetic Resonance in Medicine*, 40(2):249–260, 1998.

[42] Deng Cai, Xiaofei He, and Jiawei Han. Document clustering using locality preserving indexing. *Knowledge and Data Engineering, IEEE Transactions on*, 17(12):1624–1637, 2005.

[43] Pedro Galeano and Daniel Peña. Multivariate analysis in vector time series. 2001.

[44] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 406–417. VLDB Endowment, 2002.

[45] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.

[46] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[47] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[48] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2003.

[49] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st international conference on Machine learning*, page 11. ACM, 2004.

[50] Padhraic Smyth. Clustering sequences with hidden markov models. *Advances in neural information processing systems*, pages 648–654, 1997.

[51] Antonello Panuccio, Manuele Bicego, and Vittorio Murino. A hidden markov model-based approach to sequential data clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 734–743. Springer, 2002.

[52] Daniil Ryabko and Jérémie Mary. Reducing statistical time-series problems to binary classification. In *Advances in Neural Information Processing Systems*, pages 2060–2068, 2012.

[53] Azadeh Khaleghi, Daniil Ryabko, Jérémie Mary, and Philippe Preux. Consistent algorithms for clustering time series. *Journal of Machine Learning Research*, 17(3):1–32, 2016.

[54] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.

[55] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(2):218–233, 2003.

[56] Ali Jalali, Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering partially observed graphs via convex optimization. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1001–1008, 2011.

[57] Guangcan Liu, Zhouchen Lin, and Yong Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on machine learning*, pages 663–670, 2010.

[58] Mahdi Soltanolkotabi, Ehsan Elhamifar, Emmanuel J Candes, et al. Robust subspace clustering. *The Annals of Statistics*, 42(2):669–699, 2014.

[59] Yu-Xiang Wang and Huan Xu. Noisy sparse subspace clustering. *Journal of Machine Learning Research*, 17(12):1–41, 2016.

[60] Chao Qu and Huan Xu. Subspace clustering with irrelevant features via robust dantzig selector. In *Advances in Neural Information Processing Systems*, pages 757–765, 2015.

[61] Kenji Yamanishi and Jun-ichi Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *KDD*, 2002.

[62] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *IMC*, 2003.

[63] Yoshinobu Kawahara and Masashi Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *ICDM*, pages 389–400. SIAM, 2009.

[64] Marcel Bosc, Fabrice Heitz, Jean-Paul Armspach, Izzie Namer, Daniel Gounot, and Lucien Rumbach. Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. *Neuroimage*, 20(2), 2003.

[65] Jaxk Reeves, Jien Chen, Xiaolan L Wang, Robert Lund, and Qi Qi Lu. A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*, 46(6):900–915, 2007.

[66] Richard J Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. Image change detection algorithms: a systematic survey. *IEEE transactions on image processing*, 14(3):294–307, 2005.

[67] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *UbiComp*, pages 312–321. ACM, 2008.

[68] Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *KDD*, pages 613–618. ACM, 2003.

[69] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *WWW*, 2008.

[70] Qi He, Kuiyu Chang, and Ee-Peng Lim. Analyzing feature trajectories for event detection. In *SIGIR*, pages 207–214. ACM, 2007.

[71] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, 2010.

[72] Alexander Ihler, Jon Hutchins, and Padhraic Smyth. Adaptive event detection with time-varying poisson processes. In *KDD*, pages 207–216. ACM, 2006.

[73] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB*, pages 187–198. VLDB Endowment, 2006.

[74] Markos Markou and Sameer Singh. Novelty detection: a reviewpart 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[75] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[76] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[77] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

[78] Johan Himberg, Kalle Korpiaho, Heikki Mannila, Johanna Tikanmaki, and Hannu TT Toivonen. Time series segmentation for context recognition in mobile devices. In *ICDM*, pages 203–210. IEEE, 2001.

[79] Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn Keogh. Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In *KDD*, pages 49–58. ACM, 2015.

[80] Aristides Gionis and Heikki Mannila. Finding recurrent sources in sequences. In *Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 123–130. ACM, 2003.

[81] Seyedjamal Zolhavarieh, Saeed Aghabozorgi, and Ying Wah Teh. A review of subsequence time series clustering. *The Scientific World Journal*, 2014, 2014.

[82] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *KDD*, 2017.

[83] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2), June 2002.

[84] Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[85] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[86] W James Kent. Blatthe blast-like alignment tool. *Genome research*, 12(4):656–664, 2002.

[87] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, et al. Full-length transcriptome assembly from rna-seq data without a reference genome. *Nature biotechnology*, 29(7):644, 2011.

[88] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 2, 2004.

[89] T. S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1, 1973.

[90] Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[91] R.M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[92] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, 2008.

[93] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[94] David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

[95] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[96] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler. sMAP: a simple measurement and actuation profile for physical information. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, 2010.

[97] Scikit-learn. `http://scikit-learn.org/`.

[98] Jorge M. Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, ICANN '09, Berlin, Heidelberg, 2009. Springer-Verlag.

[99] Yiming Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2), May 1999.

[100] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[101] Carl Edward Rasmussen. The infinite gaussian mixture model. In *NIPS*, volume 12, pages 554–560, 1999.

[102] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

[103] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[104] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

[105] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[106] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118. ACM, 2001.

[107] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[108] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999.

[109] Gartner Inc. `http://www.gartner.com/newsroom/id/2636073`.

[110] Emmanuel Candes and Terence Tao. The dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, pages 2313–2351, 2007.

[111] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[112] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[113] Fang Han, Huanran Lu, and Han Liu. A direct estimation of high dimensional stationary vector autoregressions. *Journal of Machine Learning Research*, 16:3115–3150, 2015.

[114] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, pages 849–856, 2001.

[115] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[116] Sumanta Basu, George Michailidis, et al. Regularized estimation in sparse high-dimensional time series models. *The Annals of Statistics*, 43(4):1535–1567, 2015.

[117] Mahdi Soltanolkotabi and Emmanuel J Candes. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, pages 2195–2238, 2012.

[118] Pablo Montero and José A Vilar. Tsclust: An r package for time series clustering.

[119] Yudong Ma, Francesco Borrelli, Brandon Hencey, Brian Coffey, Sorin Bengea, and Philip Haves. Model predictive control for the operation of building cooling systems. *IEEE Transactions on control systems technology*, 20(3), 2012.

[120] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics, 1997.

[121] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[122] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[123] Dan Preston, Pavlos Protopapas, and Carla Brodley. Event discovery in time series. In *ICDM*, pages 61–72. SIAM, 2009.

[124] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274. ACM, 2001.

[125] LR Ford and DR Fulkerson. Maximal flow through a network. In *Classic papers in combinatorics*, pages 243–248. Springer, 2009.

[126] James M Lucas and Michael S Saccucci. Exponentially weighted moving average control schemes: properties and enhancements. *Technometrics*, 32(1):1–12, 1990.

[127] Melanie Mitchell. *An introduction to genetic algorithms*. 1998.

[128] Zach Shelby and Carsten Bormann. *6LoWPAN: The wireless embedded Internet*, volume 43. John Wiley & Sons, 2011.

[129] Project Haystack. `http://project-haystack.org/`, 2014. last visited: 05-20-2018.

[130] Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. Standard, buildingSMART, April 2014.

[131] Laura Daniele, Frank den Hartog, and Jasper Roes. Created in close interaction with the industry: the smart appliances reference (saref) ontology. In *International Workshop Formal Ontologies Meet Industries*, pages 100–112. Springer, 2015.

[132] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. Brick: Towards a unified metadata schema for buildings. In *BuildSys*, 2016.

[133] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. Brick: Metadata schema for portable smart building applications. *Applied Energy*, 2018.

[134] Anika Schumann, Joern Ploennigs, and Bernard Gorman. Towards automating the deployment of energy saving approaches in buildings. In *BuildSys*, 2014.

[135] Dezhi Hong, Hongning Wang, and Kamin Whitehouse. Clustering-based active learning on sensor type classification in buildings. In *CIKM*, 2015.

[136] Arka A Bhattacharya, Dezhi Hong, David Culler, Jorge Ortiz, Kamin Whitehouse, and Eugene Wu. Automated metadata construction to support portable building applications. In *BuildSys*, pages 3–12. ACM, 2015.

[137] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *BuildSys*, pages 13–22. ACM, 2015.

[138] Jason Koh, Dhiman Sengupta, Julian McAuley, Rajesh Gupta, Bharathan Balaji, and Yuvraj Agarwal. Scrabble: converting unstructured metadata into brick for many buildings. In *BuildSys*, page 48. ACM, 2017.

[139] Jason Koh, Bharathan Balaji, Vahideh Akhlaghi, Yuvraj Agarwal, and Rajesh Gupta. Quiver: Using control perturbations to increase the observability of sensor data in smart buildings. *arXiv preprint arXiv:1601.07260*, 2016.

[140] Weimin Wang, Michael R Brambley, Woohyun Kim, Sriram Somasundaram, and Andrew J Stevens. Automated point mapping for building control systems: Recent advances and future research needs. *Automation in Construction*, 85, 2018.

[141] Srinivas Katipamula, Ronald M Underhill, James K Goddard, Danny J Taasevigen, MA Piette, J Granderson, Rich E Brown, Steven M Lanzisera, and T Kuruganti. Small-and medium-sized commercial building monitoring and controls needs: A scoping study. Technical report, Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2012.

[142] Arka Bhattacharya, Joern Ploennigs, and David Culler. Short paper: Analyzing metadata schemas for buildings: The good, the bad, and the ugly. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 33–34. ACM, 2015.

[143] Jonathan Fürst, Kaifei Chen, Randy H Katz, and Philippe Bonnet. Crowd-sourced bms point matching and metadata maintenance with babel. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[144] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[145] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[146] Dezhi Hong, Hongning Wang, Jorge Ortiz, and Kamin Whitehouse. The building adapter: Towards quickly applying building analytics at scale. In *Proceedings of the 2nd ACM Conference on Embedded Systems for Energy-Efficient Built Environments*, BuildSys'15, 2015.

[147] Dezhi Hong, Quanquan Gu, and Kamin Whitehouse. High-dimensional time series clustering via cross-predictability. In *AISTATS*, pages 642–651, 2017.

[148] Peter Bailis, Kunle Olukotun, Christopher Ré, and Matei Zaharia. Infrastructure for usable machine learning: The stanford DAWN project. *CoRR*, abs/1705.07538, 2017.

[149] Nicolas Hug. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

[150] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. Openrec: A modular framework for extensible and adaptable recommendation algorithms. In *WSDM*, pages 664–672, New York, NY, USA, 2018. ACM.

[151] Christian Beckel, Wilhelm Kleiminger, Romano Cicchetti, Thorsten Staake, and Silvia Santini. The eco data set and the performance of non-intrusive load monitoring algorithms. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 80–89. ACM, 2014.

[152] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. Nilmtk: an open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th international conference on Future energy systems*, pages 265–276. ACM, 2014.

[153] Arctic. `https://github.com/manahl/arctic`, 2014. last visited: 05-20-2018.

[154] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[155] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh a python package). *Neurocomputing*, 2018.

[156] Jingkun Gao and Mario Bergs. A large-scale evaluation of automated metadata inference approaches on sensors from air handling units. *Advanced Engineering Informatics*, 37:14 – 30, 2018.

[157] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[158] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[159] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[160] BACnet-A Data Communication Protocol for Building Automation and Control Networks. Standard, ASHRAE, GA, USA, 2016.

[161] Dezhi Hong, Hongning Wang, and Kamin Whitehouse. Clustering-based active learning on sensor type classification in buildings. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '15, 2015.

[162] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.

[163] Turtle. `https://www.w3.org/TR/turtle/`.

[164] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

[165] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill 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*, DMKD '03, 2003.

[166] Lexiang Ye and Eamonn Keogh. Time series shapelets: A new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, 2009.

[167] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. Clustering time series using unsupervised-shapelets. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, 2012.

[168] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[169] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[170] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[171] Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.

[172] Martin J Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using-constrained quadratic programming (lasso). *Information Theory, IEEE Transactions on*, 55(5):2183–2202, 2009.

[173] René Brandenberg, Abhi Dattasharma, Peter Gritzmann, and David Larman. Isoradial bodies. *Discrete & Computational Geometry*, 32(4):447–457, 2004.