The Building Adapter: Towards Quickly Applying Building Analytics at Scale

Dezhi Hong[1], Hongning Wang[1], Jorge Ortiz[2], Kamin Whitehouse[1]
[1]Department of Computer Science, University of Virginia, VA USA
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY USA
{hong, hw5x, whitehouse}@virginia.edu, jjortiz@us.ibm.com

ABSTRACT

Building analytics can produce substantial energy savings in commercial buildings by automatically detecting wasteful or incorrect operations. However, a new building's sensing and control points need to be mapped to the inputs of an analytics engine before analysis is feasible and the process of *mapping* is a highly manual process - a key obstacle to scaling up building analytics. In this paper, we present new techniques to perform automatic mapping *without* any manual intervention. Our approach builds on and improves upon techniques from *transfer learning*: it learns a set of statistic classifiers of the metadata from a labeled building and adaptively integrates those classifiers to another unlabeled building, even if the two buildings have very different metadata conventions. We evaluate this approach using 7 days' data from over 2,500 sensors located in 3 commercial buildings. Results indicate that this approach can automatically label at least 36% of the points with more than 85% accuracy, while the best baseline achieves only 63% label accuracy on average. These techniques represent a first step towards technology that would enable any new building analytics engine to scale quickly to the 10's of millions of commercial buildings across the globe, without the need for manual mapping on a per-building basis.

Categories and Subject Descriptors

C.3 [**Special-Purpose And Application-Based Systems**]: Real-time and embedded systems

General Terms

Performance, Experimentation

Keywords

Building, Sensor Type, Transfer Learning

BuildSys'15, November 4–5, 2015, Seoul, South Korea..

DOI: http://dx.doi.org/10.1145/2821650.2821657.

1. INTRODUCTION

To incentivize the reduction of building energy consumption, the U.S. government launched the Better Buildings Challenge to make buildings at least 20 percent more efficient by 2020 [13]. To achieve this goal, many organizations are applying data analytics to the thousands of sensing and control points[1] in buildings to detect wasteful, incorrect, and inefficient operations. Many promising analytical approaches have been created that demonstrate promise for substantial energy savings [14, 8]. However, these analytic engines are *tightly coupled* to the database schemata and metadata conventions in the buildings for which they were designed. In practice, unfortunately, most buildings use different metadata conventions depending on the type of equipment in the building, the vendors of the equipment, and the contractors who originally installed it. Therefore, the same analytics cannot easily be applied to different buildings where the type, location, and relationships between sensors are represented differently, before those different schemata and conventions are mapped to a common norm. 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 equipment installation. It can take days or weeks and is a key obstacle to the widespread use of building analytics.

Several solutions have been proposed to facilitate the mapping problem. Bhattarcharya et al. [3] formulate a solution that derives a set of regular expressions from a handful of labeled examples to normalize the sensor point names, where the example selection is aided by data feature-based clustering. Schumann et al. [26] develop a probabilistic framework to classify sensor types based on the similarity between a point name and the entries in a manually constructed dictionary. Hong et al. [17] propose an active learning based approach to iteratively acquire human labels for informative point names for supervised statistical classifier learning and propagate the acquired labels to similar points. 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.

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 affected by a given air handler unit. These

[1]A sensing or control "point" is a sensor, a controller, or a software value.

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.

In this paper, we envision a technology that would enable building analytics engine to be quickly 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. The key to this vision is to perform all mapping operations *automatically* and to remove the human from the scalability equation entirely. Any such system would not need to perform perfect mapping; it must only do a first pass to label an initial batch of important points (e.g., room temperature) with decent accuracy that is good enough to flag the right buildings for a more detailed, manual inspection process.

As our contribution, we take a first step towards scalable building analytics by developing new techniques to automatically infer the *sensor type* information in a building *without* manual labelling. The 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 but might not be consistent across buildings, while the sensor readings are more consistent across buildings but are likely to be poor indicators of sensor types. Our techniques combine the complementary strengths of these two attributes to automatically recognize sensor types in one building based on another 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 time-series data. These classifiers predict sensor types based on the raw sensor 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 label the points in the second unlabeled 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.

To evaluate our solution, we use a dataset[2] with 7 days of data from over 2,500 sensors located in 3 commercial buildings across 2 different college campuses, and true sensor type was created manually for all three buildings. Then, we applied our techniques on each building to automatically infer the sensor type for the other two buildings. The experimental results indicate that the proposed solution can automatically label at least 36% of the points with more than 85% accuracy, and in some cases labels up to 81% of the points with 96% accuracy. In contrast, training classifiers on one building and applying them directly to another building achieves only 63% label accuracy on average. Our technique does not label all points, but those that are labeled have near perfect label accuracy. These labeled points usually belong to the most common types such as temperature and CO_2, which can form an important base for further analysis. Before concluding, we present two techniques that are showing promise to improve this coverage and expand support for a larger fraction of analytics algorithms.

[2]This dataset was simply employed as a testing case and the development of our technique was data- agnostic without any customization for this dataset.

2. BACKGROUND AND RELATED WORK

In modern commercial buildings, metadata is often expressed as short text strings with several concatenated abbreviations in a point name. Table 1 lists a few point names of sensors from three different building management systems (Trane[3], Siemens[4] and Barrington Controls[5]). For example, the point name `SODA1R300__ART` is constructed as a concatenation of the building name (`SOD`), the air handler 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 control unit that can affect the temperature in this room. Clearly, different naming conventions are used in these buildings. For example, the notion of *room temperature* is encoded with a different abbreviation in each of the three buildings: `Temp`, `RMT` and `ART`. Such variations across different buildings impose great difficulty in quickly 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 1: Example point names for temperature sensors from three different buildings.

To the best of our knowledge, we are the first to develop transfer learning based solutions to address the problem of automated sensor type classification across buildings.

Researchers have tried to systematically address the problem of point name normalization. Dawson-Haggerty et al. [12] and Krioukov et al. [21] introduce 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. Bhattarcharya et al. [3] exploit a programming language based solution, where they derive a set of regular expressions from a handful of labeled examples to normalize the point name of sensors. This approach assumes a consistent format for all point names across buildings, which might not be true in practice (as shown in Table 1). Schumann et al. [26] develop 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 a lot of variations of the same type, or conflicting definitions of a dictionary entry in differ-

[3]`http://www.trane.com/`

[4]`http://www.siemens.com/`

[5]The company is no longer in business.

ent buildings. Hong et al [17] formulate an active learning based approach to iteratively acquire human labels for informative examples and propagate the acquired labels among points. However, all aforementioned work depends on manual annotations, and thus none of them address the scalability issue of metadata normalization across buildings nor leverage the knowledge from already labeled buildings.

Applying transfer learning to cross building sensor type classification saves extra effort in manual annotation by exploiting the labels in the already well labeled buildings. There are several categories of transfer learning, e.g., inductive, transductive, and multi-task transfer learning as comprehensively surveyed in [24]. Inductive transfer learning [9] 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 [6] has a similar setting, but tries to learn from the target and source domains simultaneously. Transductive transfer learning [10] 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 reweight the source domain trained classifiers' predictions in the target domain, e.g., instance-based local weighting [4, 19, 29]. 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 [1, 20]. Our problem setting falls into this category: we assume we have well-labeled instances from one source building, but do not have any labeled instances in the target building. We exploit different properties of a sensor point to perform the transfer learning: sensor's data is utilized to estimate a diverse set of classifiers to transfer knowledge from the source building to target building; sensor names in the target building are used to compute the ensemble weight of classifiers during knowledge transfer.

3. THE BUILDING ADAPTER

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 for classifying sensor types in a new building. Specifically, we construct classifiers based on 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 applied to a new target building, since different classifiers might be effective in predicting different instances, we appeal to the instance-specific local weighting method [16] to weight those 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 on 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.

3.1 Feature Representation

Our solution 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 in differentiating sensor types. In this section, we describe the construction of two different sets of features, i.e., data features and name features.

3.1.1 Data Features

A signal[6] 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 demonstrated in Figure 1. Similar to piecewise aggregated approximation (PAA [7]) – where the mean is calculated in a fixed-length window – we compress the signal by computing a set of summary statistics over fixed-length windows. Table 2 summarizes 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 2: Statistical functions applied to each time-series on window level.

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 above. 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:

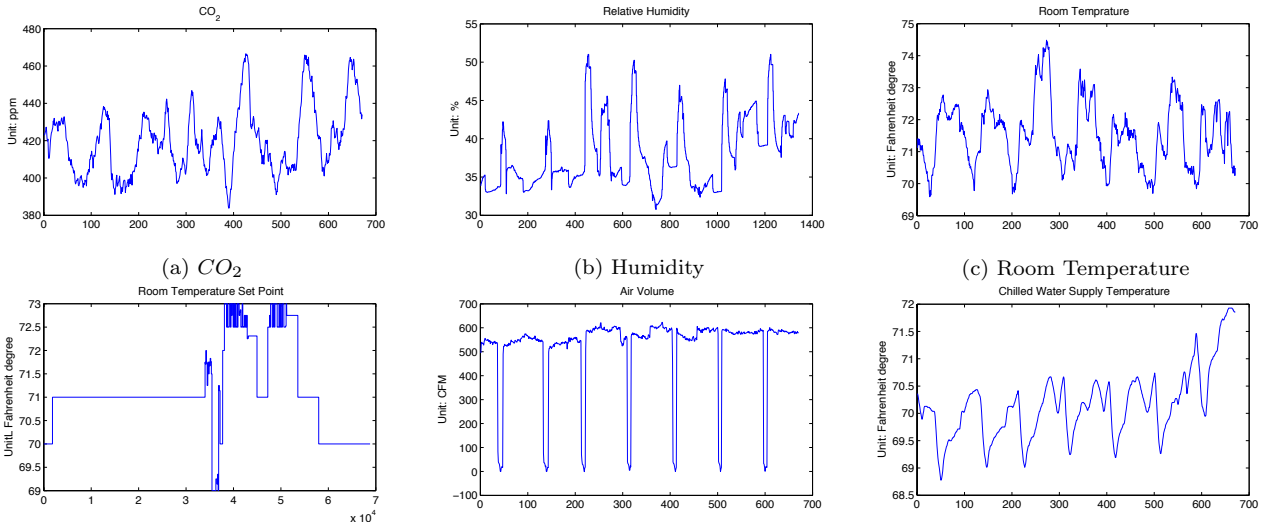
$$\begin{aligned} F = \{&min(MIN), max(MIN),\\ &median(MIN), var(MIN),\\ &...\\ &min(SLOPE), max(SLOPE),\\ &median(SLOPE), var(SLOPE)\} \end{aligned}$$

F is the data feature representation of each sensor stream used in our study.

3.1.2 Name Features

Sensor point names are short text strings with several concatenated abbreviations, as shown in Table 1. To extract

[6]In this paper, we use the term "signal", "trace" and "time-series" *interchangeably*.



(a) CO_2 (b) Humidity (c) Room Temperature

(d) Room Temperature Set Point (e) VAV Air Volume (f) Chilled Water Supply Temperature

Figure 1: Different types of sensor generally have different amplitudes that can be separated and binned to characterize the data.

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 [22], which helps measure sequence similarity without requiring alignment, to capture variations in type abbreviations. For example, the type of "temperature" might be encoded as "tmp" or "temp" and only considering the bag-of-words 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 and we limit our k-mers computation within a word boundary. For example, for a string "ABCDEF", the 3-mers (k=3) generated would be `{ABC, BCD, CDE, DEF}`.

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 Locally Weighted Ensembles for Knowledge Transfer

To effectively encapsulate and transfer knowledge from one labeled building to another, we construct a set of different classifiers, including support vector machines [30], logistic regression [18] and random forest [5], 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 paper, 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 [16] 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 a base classifier's 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 be the data feature vector of an instance 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,

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

where $p(y|x, D_T, M_i) = p(y|x, M_i)$, because $x \in D_T$ and $p(y|x, M_i)$ 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 $x \in D_T$, $p(M_i|D_T)$ equals to $p(M_i|x)$, which is the locally adjusted weight for M_i, and Eq. 1 becomes,

$$p(y|x) = \sum_{i=1}^{k} w_x^{M_i} p(y|x, M_i) \tag{2}$$

where $w_x^{M_i} = p(M_i|x)$. 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 the estimated neighborhood. We will next explain how the weight is calculated per instance.

3.3 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 [16]: 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 on the area surrounding x have greater consistency with the clustering results, M_i will be assigned a larger weight for 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 point 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. The neighborhood graph describes whether different examples belong to the same class/cluster. In the graph constructed for 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 be assigned a larger weight in the ensemble. So the weight $w_x^{M_i}$ for M_i at x is proportional to the similarity of the two graphs:

$$w_x^{M_i} \propto s(G_M, G_C|x) = \frac{|V_M \cap V_C|}{|V_M \cup V_C|} \tag{3}$$

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 two graphs. Figure 2 illustrates an example of neighborhood graphs for an instance x (in grey circle): two different classifiers predict different instances (in white circles) as neighbors to the target instance (in grey circles), where 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.

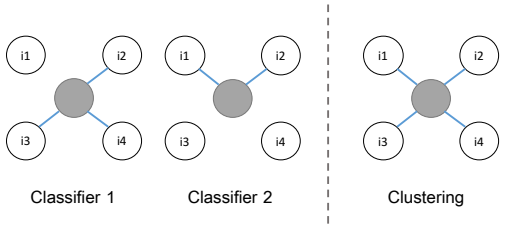


Figure 2: An example of local neighborhood graphs of x (in grey circle): two different classifiers predict different instances as neighbors to the target instance, where classifier 1's prediction is more similar than classifier 2's to the cluster structure.

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

$$w_x^{M_i} = \frac{s(G_{M_i}, G_C|x)}{\sum_{i=1}^{k} s(G_{M_i}, G_C|x)} \tag{4}$$

The final prediction for x is simply $\hat{y} = argmax_y \ p(y|x)$ as $p(y|x)$ is defined in Eq. 2.

3.3.1 Distance-based Adjustment

In some cases the similarity score defined in Eq 4 can be problematic. Consider the case shown in Figure 3. The example shows two classifiers where the target instance has two neighbors. The distance between the instances are 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 higher weight. To fix the issue we include the distance between instances into consideration and adjust similarity score as,

$$s^*(G_M, G_C|x) = 1 - \frac{\sum d_{V_I}/|V_I|}{\sum d_{V_U}/|V_U|} \tag{5}$$

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 $\sum d_{V_U}$).

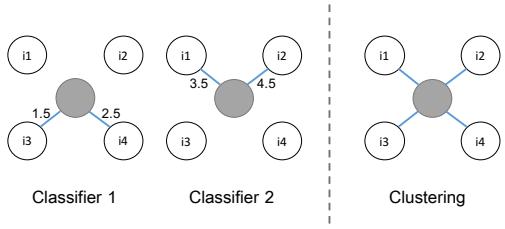


Figure 3: An example of local neighborhood graphs of x with the distance between instances into consideration.

3.3.2 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,

$$\bar{s}_x = \frac{1}{k} \sum_{i=1}^{k} s(G_{M_i}, G_C|x) \tag{6}$$

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 section.

3.4 Non-parametric Bayesian Clustering

In our transfer learning based approach, clustering structure for instances in target building is exploited to approximate the true local structure of an instance. However, if employing clustering algorithms such as Gaussian Mixture Model (GMM) [32], we need to manually specify the number of clusters for a given input data set; and the clustering results are very sensitive to this setting. More importantly, 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. It is impossible for us to pre-specify the optimal number of clusters. To make clustering feasible on the new target building, we use a non-parametric Bayesian solution as in [17] - Dirichlet Process with GMM, which automatically decides the number and structure of clusters. In particular, name feature distribution $p(x)$ is exploited via its latent clustering structure.

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 estimate, we choose the isotropic Gaussian in our solution,

$$p(x|c) = (2\pi\sigma^2)^{-d/2} \exp -\frac{(x-\mu_c)^{\mathsf{T}}(x-\mu_c)}{2\sigma^2} \qquad (7)$$

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.

We assume the model parameters (α, μ) in each cluster are also random variables, which are drawn from a Dirichlet Process prior [15]. A Dirichlet Process $DP(G_0, \eta)$ with a base distribution G_0 and a scaling parameter η is a distribution over distributions [15]. 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 the 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 the data. The number of unique draws, i.e., the number of clusters, varies with respect to the data and therefore is random, instead of being pre-specified.

As a result, with the $DP(G_0, \eta)$ prior, the data density in a given collection of instances can be expressed using a stick-breaking representation [28]:

$$p(x) = \sum_{c=1}^{\infty} \alpha_c \mathcal{N}(x|\mu_c, \sigma) p(\mu_c|G_0) \qquad (8)$$

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 (8) 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. Following the sampling scheme proposed in [23], we use a Gibbs sampling method to infer the posterior of cluster membership. Detailed specifications of this sampling algorithm can be found in [23].

Putting it all together: Algorithm 1 summarizes our transfer learning algorithm for the problem of sensor type classification across buildings. We start by 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 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 large 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 classifier and obtain the label y for x.

Algorithm 1: Transfer Learning for Sensor Type Classification

Input: Data features of the source building $\mathcal{D}_S = \{x_1^D, x_2^D, \ldots, x_n^D\}$ and their labels $\mathcal{Y}_S = \{y_1, y_2, \ldots, y_n\}$ data features of the target building $\mathcal{D}_T = \{x_1^D, x_2^D, \ldots, x_m^D\}$, and name features of the target building $\mathcal{P}_T = \{x_1^P, x_2^P, \ldots, x_m^P\}$
Output: predicted labels of the instances in target building $\mathcal{Y}$
Initialize: Generate clusters with $DP(G_0, \eta)$ on name features in $\mathcal{P}_T$
Train k classifiers $M_1, \ldots, M_k$ based on $\mathcal{D}_S$ and $\mathcal{Y}_S$;
for x^D *in* $\mathcal{D}_T$ **do**
  Construct neighborhood graphs G_M and G_C for x^D as defined in Section 3.3 for each M_i;
  Compute the similarity score for each M_i with Eq. 5;
  Check the average similarity score $\hat{s}_x$ over all M_i with Eq. 6;
  If $\hat{s}_x > \delta$, then use Eq. 4 and Eq. 2 to predict the label y;
end

4. EVALUATION

To investigate the effectiveness of our solution, we evaluate our transfer learning based classification technique on the sensor reading data and the point names of sensors from three commercial buildings. Extensive experiments demonstrate that our technique is able to accurately classify sensor types for a considerable fraction of the instances without any human intervention. With more training data, the performance of our technique can be further enhanced. To demonstrate the promising usage of the technique, we combine transfer learning with traditional labeling technique to accelerate the learning efficiency.

4.1 Taxonomy and Data Collection

Table 3 summarizes all the sensor types evaluated in our experiments in these three buildings and the number of sensors for each type. For example, "room temperature" measures the temperature in room, and the other temperature measurements for water circulation and air ventilation are

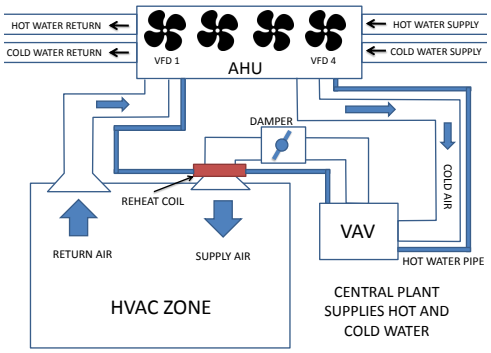


Figure 4: A typical HVAC system consisting of an air handler unit (AHU), several variable air volume boxes (VAV), water-based heating/cooling pipes and air circulation ducts. (Figure used with permission from the authors of [2].)

illustrated in Figure 4. Setpoints are grouped together into one general type.

Our evaluation dataset, which contains both sensor reading data and point names of sensor streams, is collected from over 2,500 sensors of different types deployed in three commercial buildings. We collected a week's sensor reading data from each building. Building A is Rice Hall at the University of Virginia, where the sense points report to a database component in a Trane BMS, between every 10 seconds to 10 minutes. Both building B and C are from UC Berkeley: B is Sutardja Dai Hall, which contains sensors and equipment from KETI[7] and Siemens. Building C is Soda Hall, which that uses an archaic system by Barrington Controls which is no longer in business. Points and sensors in these two buildings transmit data to an sMAP [11] archiver periodically between every 5 seconds to 10 minutes.

All of our learning and classification processes are implemented with the scikit-learn [27] library; an open-source machine learning package implemented in Python.

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 data features to be more generalizable than point names since building environments are conditioned similarly while the sensor naming conventions could 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 illustrated earlier in Table 1.

To examine how well each type of features can be used for knowledge transfer, we perform sensor type classification across buildings with each set 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 4 [8] summarizes the results,

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

[8] We also performed this experiment with other statistical classifiers, but due to space limit we only reported the results by random forest and the choice of classifiers does not affect the conclusion. Besides, we performed feature selection for classification and the performance differences between different sets of features are marginal.

Type	Building 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: Number of points by type for the 3 test buildings. "Temp" stands for "temperature", "HW" for "hot water" and "CW" for "cold water".

	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 4: 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.

from which we can conclude that data features are more useful to build a classifier across buildings than name features, but the resulting performance is not perfect. These observations confirm our assumption that data feature is preferable to train statistic classifiers across buildings.

4.3 Features for Clustering

As shown in Algorithm 1, 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 aligned with their true class labels: the higher quality of clusters is, the more accurate the weights will be given to the base classifiers.

Table 5 shows the quality of clusters generated by our non-parametric Bayesian method on data features and name features. Cluster quality is measured by rand index [25], which

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

Table 5: Quality of clusters generated in three buildings with different features measured by rand index (in the range [0,1], higher is better).

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 constructed by 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 name features to estimate local clustering structures in target buildings.

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 selected 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 summarized in Table 6. 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 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.

4.5 Transfer Learning Performance

We first consider the case where only one building is exploited as the source, i.e., all 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 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 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.

As an example, Figure 6 presents a confusion matrix for 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 in each sensor type.

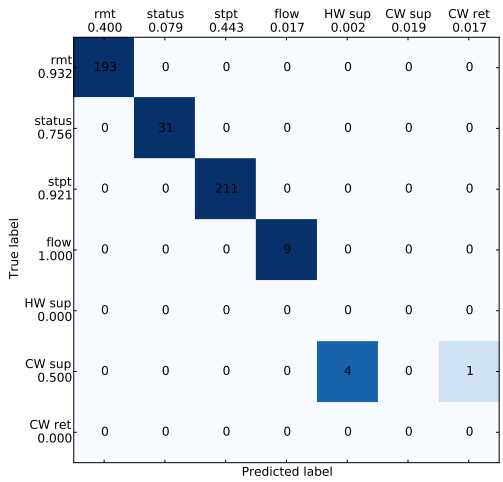


Figure 6: Confusion matrix for the classification results of transferring from A to C. "rmt" stands for room temperature, "stpt" for setpoint, "sup" for supply, "ret" for return, and "SAT" for supply air temperature.

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 [31], 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).

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->B). The average performance is reported in Table 7. 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 other cases 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.

5. FUTURE DIRECTIONS

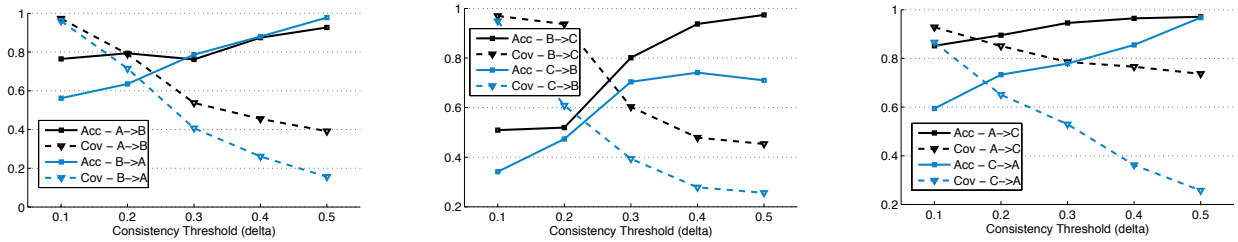
The results above indicate our proposed solution can label a fraction of the points in a commercial building with no human intervention. As such, it will serve some but not all forms of building analytics, depending on which sensing points they need. In this section, we outline several promising directions to expand these results to a larger fraction of the building points.

	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 6: Base classifier performance across buildings on data features: each cell contains the accuracy for random forest, logistic regression and SVM, respectively. The accuracy varies between 0.158 and 0.921, with random forest being the best performer.

	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 7: Accuracy and F1 score of transfer learning and the best base classifier (random forest) with the threshold δ set to be 0.4. Each cell contains two numbers for our approach and the baseline, respectively. Overall, our approach outperforms the baseline with respect to both criteria.



(a) A and B (b) B and C (c) A and C

Figure 5: Type classification accuracy (Acc) against labeled percentage (Cov) with transfer learning between different pairs of buildings (denoted as X->Y). As we increase the threshold, the coverage drops while the overall accuracy increases.

5.1 Training on Multiple Buildings

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

In practice, it is common to have labels for multiple buildings. Combining them as the single source could lead to 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 ten runs is reported. From Table 8, we see that combining multiple buildings as the source is superior to single source.

Upon further inspection of results, 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 with labeled data from multiple buildings produces more robust classifiers and as hundreds of buildings are normalized, this approach will become even more promising. In contrast, existing approaches scale only linearly with the number of buildings being normalized since they all require manual labeling and only apply to a single building.

5.2 Complementing Manual Labeling

Another important question we want to answer is how much our transfer learning based approach can expedite traditional labeling techniques. Since our technique automatically labels a considerable percentage of sensor points in a building as a first step, the manual labeling process in this new building can be accelerated. We adopt the active learning technique developed in [17] and examine how much our technique can accelerate the labeling process. For brevity, we refer interested reader to [17] for the detailed algorithm.

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 automatic labeling process with a $\delta = 0.6$ to minimize the amount of inaccurate labels from transfer learning. Then we switch to the AL algorithm developed in [17] where we acquire one manual label per iteration. Due to space limit, we only show the results from transferring from building B to building A in Figure 7. We observe that AL+TL helps reduce the number of manual labels to achieve the same performance as AL. In other words, given the same amount of manual labels, AL+TL can always outperform AL. As il-

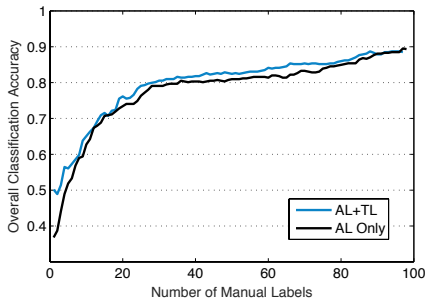


Figure 7: Our transfer learning based approach can complement traditional labeling technique to achieve better accuracy with the same amount of labels.

lustrated earlier, more training sources could further boost the performance of transfer learning, which we expect would also significantly expedite traditional labeling techniques for a single building.

6. CONCLUSION

In this paper, we take a first step towards scalable building analytics by developing new techniques to automatically infer the type information in a building *without* manual labeling. Our techniques combine the complementary strengths of the data and the point name of a sensor to automatically create metadata for one building based on another 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 labels in a new building with at least 85% accuracy, and for some cases up to 81% with more than 96% accuracy. As future steps, we illustrate how combining multiple buildings as the training source could further boost the performance, and how our solution can complement traditional labeling technique. The solution is general, simple yet effective, which we believe can act as a tool for metadata construction for not only building sensors, but also in the broader context of the Internet of Things.

7. ACKNOWLEDGEMENTS

Thanks to our shepherd, Jin Wen, and the anonymous reviewers for helpful comments. This work was funded in part by NSF awards 1305362, 1038271, and 0845761.

8. REFERENCES

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artif. Intell. Rev.*, 11(1-5), Feb. 1997.

[2] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal. Sentinel: Occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *SenSys '13*, 2013.

[3] A. Bhattacharya, D. Hong, D. E. Culler, J. Ortiz, K. Whitehouse, and E. Wu. Automated metadata construction to support portable building applications. In *BuildSys'15*, 2015.

[4] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *ICML '07*, 2007.

[5] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[6] R. Caruana. Multitask learning. *Machine Learning*, 28(1), 1997.

[7] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2), June 2002.

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

[9] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *ICML '07*, 2007.

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

[11] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. sMAP: a simple measurement and actuation profile for physical information. In *SenSys '10*, 2010.

[12] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler. Boss: Building operating system services. In *NSDI'13*, 2013.

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

[14] B. Dong and K. Lam. A real-time model predictive control for building heating and cooling systems based on the occupancy behavior pattern detection and local weather forecasting. *Building Simulation*, 7(1), 2014.

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

[16] J. Gao, W. Fan, J. Jiang, and J. Han. Knowledge transfer via multiple model local structure mapping. In *KDD '08*, 2008.

[17] D. Hong, H. Wang, and K. Whitehouse. Clustering-based active learning on sensor type classification in buildings. In *CIKM '15*, 2015.

[18] D. W. Hosmer Jr and S. Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

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

[20] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1), Mar. 1991.

[21] A. Krioukov, G. Fierro, N. Kitaev, and D. Culler. Building application stack (bas). In *BuildSys '12*, 2012.

[22] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

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

[24] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22, Oct 2010.

[25] J. M. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *ICANN '09*, Berlin, Heidelberg, 2009. Springer-Verlag.

[26] A. Schumann, J. Ploennigs, and B. Gorman. Towards automating the deployment of energy saving approaches in buildings. In *BuildSys '14*, 2014.

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

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

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

[30] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

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

[32] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *ICPR'04*, volume 2, 2004.