Scrabble: Transferrable Semi-Automated Semantic Metadata Normalization using Intermediate Representation

Jason Koh
University of California, San Diego
jbkoh@eng.ucsd.edu

Bharathan Balaji
University of California, Los Angeles
bbalaji@ucla.edu

Dhiman Sengupta
University of California, San Diego
dhimnsen@eng.ucsd.edu

Julian McAuley
University of California, San Diego
jmcauley@eng.ucsd.edu

Rajesh Gupta
University of California, San Diego
gupta@eng.ucsd.edu

Yuvraj Agarwal
Carnegie Mellon University
yuvraj@cs.cmu.edu

ABSTRACT

Interoperability in the Internet of Things relies on a common data model that captures the necessary semantics for vendor independent application development and data exchange. However, traditional systems such as those in building management are vertically integrated and do not use a standard schema. A typical building can consist of thousands of data points. Third party vendors who seek to deploy applications like fault diagnosis need to manually map the building information into a common schema. This mapping process requires deep domain expertise and a detailed understanding of intricacies of each building's system. Our framework - *Scrabble* - reduces the mapping effort significantly by using a multi-stage active learning mechanism that exploits the structure present in a standard schema and learns from buildings that have already been mapped to the schema. Scrabble uses conditional random fields with transfer learning to represent unstructured building information in a reusable intermediate representation. This reusable representation is mapped to the schema using a multilayer perceptron. Our novel semantic model based active learning mechanism requires only minimal input from domain experts to interpret esoteric, idiosyncratic data points. We have evaluated Scrabble on five buildings with thousands of different entities and our method outperforms prior work by 59%/162% higher Accuracy/Macro-averaged-F1 in a building when 10 examples are provided by an expert in both cases. Scrabble achieves 99% Accuracy with 100-160 examples for buildings with thousands of points while the other baselines cannot.

CCS CONCEPTS

• **Information systems → Entity resolution**; • **Computer systems organization →** *Sensors and actuators*;

KEYWORDS

smart buildings, metadata schema, machine learning

ACM Reference Format:
Jason Koh, Bharathan Balaji, Dhiman Sengupta, Julian McAuley, Rajesh Gupta, and Yuvraj Agarwal. 2018. Scrabble: Transferrable Semi-Automated Semantic Metadata Normalization using Intermediate Representation. In *The 5th ACM International Conference on Systems for Built Environments (BuildSys '18), November 7–8, 2018, Shenzen, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3276774.3276795

1 INTRODUCTION

The Internet of Things (IoT) envisions a distributed processing platform of sensors and devices, that work together to enable new applications spanning health, transport and energy systems. This vision faces many real-life challenges. Primarily, the composition of data and services provided by a diverse group of vendors. These vendors use different schema (if any at all) even when interfacing with the same or similar sets of sensors. Emerging IoT solutions like Apple Home Kit [1] and attempt to standardize such interactions. However, legacy systems in buildings lack a common information model, that helps different systems ingest each other's data.

With a general framework for combination and navigation of diverse sensory data as a long-term goal, we focus here on methods that can effectively use data collected from diverse commercial buildings. This is possible because of the use of Supervisory Control And Data Acquisition (SCADA) systems that cover over 14% of the buildings in the US [4]. These SCADA systems can have hundreds to thousands of data points, i.e., timeseries of measurements, such as those from sensors in lighting and fire safety systems.

Unfortunately, such raw data is not very useful in building operations. System operators need contextual information about a data point to identify type of sensor, the location of the sensor, etc. Such information is usually carried as "metadata" associated with actual sensor data. Metadata describing the data points can be heterogeneous and inconsistent in naming across buildings and vendors. This makes it very hard for third-party developers and vendors to navigate building information. Often a manual mapping is done by the users of sensor data, which requires significant input from the domain experts in building systems design and operations. This mapping process is a bottleneck in deploying portable applications across building systems or integrating multiple data sources for analyses. We seek to automate the mapping process with minimal or no input from domain experts by exploiting known structured information in the underlying metadata.

Prior research has framed the metadata mapping process as semi-supervised learning [6, 9, 14], multi-class labeling [12], an

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
BuildSys '18, November 7–8, 2018, Shenzen, China
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5951-1/18/11.
https://doi.org/10.1145/3276774.3276795

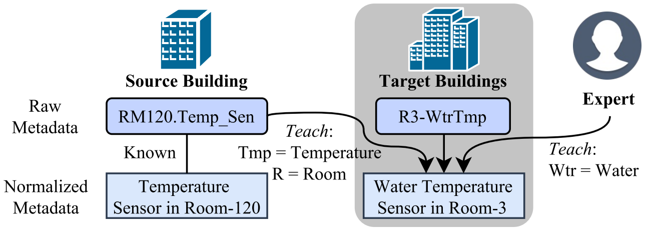


Figure 1: Fast metadata normalization of new buildings with a known building's information and an expert's knowledge.

edit distance based dictionary lookup [23], or transfer learning [13]. However, these approaches either require significant domain expert input to achieve high accuracy [6, 9, 12], or suffer from low precision and recall [13]. Both transfer learning and active learning methods lack in either precision or the human effort to achieve the precision. We need a hybrid method that can address both aspects, i.e. exploit existing mappings while augmenting the model further if needed.

We present *Scrabble*, our framework to retrieve semantic metadata from unstructured raw metadata while reusing known information in existing buildings to reduce the amount of effort for domain experts to provide input labels. Fig. 1 provides an overview of the goal that known metadata can be mapped automatically (e.g. 'Tmp' is Temperature in target buildings) and ask domain experts to provide undiscovered labels (e.g. 'Wtr' is Water). Scrabble uses a two-stage, active learning approach exploiting a known taxonomy of labels and mapping information from existing buildings. At the first stage, we learn a Conditional Random Fields (CRF) model [15] to extract reusable intermediate representations (IR) from character sequences in an existing "source building" that has already been mapped to a known schema. In the second stage, we learn a multilabel classifier to map the IR to actual labels. We use multi-layer perceptron (MLP) for the multilabel classification with its capability of handling the high dimension of the input words. For the IR, labels and the taxonomy, we use a semantic ontology called Brick [5] that specifies a list of equipment, data points and relationships between them. We use active learning methods to ask domain experts' input for the unmapped data points. Our model enables smoother transfer of mappings from known buildings to a new target building while exploiting different types of information sources systematically.

We have implemented and evaluated Scrabble on metadata from five diverse buildings on 6,551 randomly chosen data points from a total of 21,802 points. We verified the ground truth of those points manually. With the IR and proper classifiers, Scrabble extracts entities from unstructured metadata with an improvement of 59%/162% higher Accuracy/Macro-averaged-F1 in a building than a baseline with 10 initial examples. Furthermore, Scrabble can achieve 99% Accuracy with 100-160 examples for buildings with thousands of points while the baselines cannot.

2 BACKGROUND

2.1 Smart Buildings Metadata

Modern buildings have many different equipment and control systems in place to provide services like lighting, heating, and air conditioning. Much of this infrastructure is supported using sensors and networked into a building management system (BMS) for remote monitoring and operation. BMSes manages data from sensors such as room temperature, power meter; from actuators such as dampers for control of air flow, fans for exhaust; and from configurations such as the temperature setpoint for heating.

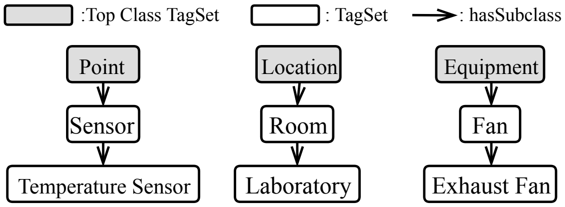


Figure 2: *Brick taxonomy example* [5]. *Point, Location* and *Equipment* are top level TagSets where the others are subclasses of them. E.g., *Server Room* is a subclass of *Room*. The max depth of the hierarchy is 6 and there are 904 TagSets.

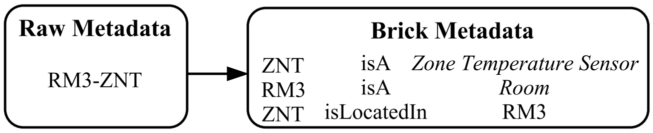


Figure 3: An example of the metadata normalization. The raw metadata, RM3-ZNT, converted to Brick in Turtle syntax. ZNT is an instance of *Zone Temperature Sensor* and RM is an instance of *Room*. isLocatedIn explains the sensor is located in the room.

Contextual information of the data are encoded as metadata and they are manually written by original system vendors. Thus, the metadata are unstructured and not standardized across different buildings or even in the same building [8]. Table 1 shows example metadata of different buildings across two campuses. BACNet [10] is a network protocol for buildings. There are predefined codes such as `BACNet Type` but most of the meaningful information is found in raw strings such as `BACNet Name, Vendor Name` and `BACNet Description`. Mainly, four types of information are written in the metadata: point type, equipment, location and network interfaces. The information is arbitrarily positioned and has many variations. For example, temperature can be shown as Tmp, Temp, TEMP, T, Tempreature, etc. It is difficult to formulate deterministic algorithms that capture all variations such as an abbreviation, single characters and even spelling errors. Whether a point is a sensor or a setpoint is also unclear as sensors are not explicitly described. Metadata can also be implicit, e.g., 3FLW-HALL in the third row of the Table represents W-HALL, a room on the third floor. BACnet's general concept of having human readable names and codified metadata is also common in other standards such as KNX.

2.2 Brick: Building Metadata Schema

Brick is a metadata schema for enabling portable applications for smart buildings [5]. It provides complete vocabularies and relationships to describe resources such as sensors and equipment in buildings in a machine-readable format. Scrabble uses Brick as target schema but it can be extended to other schemata if needed.

Table 1: Metadata examples in different buildings on different campuses. Each row in the tables is metadata for a data point in the upper table and corresponding labels in the lower table.

Campus-Building	Vendor Name	BACNet Name	BACNet Description	BACNet Type
A-1	ENG.CRAC-1.TEMPSETF	NAE 05 N2 2 VND 162 TEMPSETF	Temp Setpoint	Analog Output
A-2	SC-CRAC-1-MIG-008.Tmp	NAE 14 N2 Trunk 1 MIG 008 Temp	Temperature	Analog Input
A-2	SC.3FLW-HALL.ZN-T	NAE 13 N2 Trunk 2 VAV327 ZN T	Zone Tempreature	Analog Input
B-1	RM123A Zone Temp 3	N/A	N/A	N/A

Campus-Building	Point Label	Equipment Labels	Location Labels	Network Interfaces
A-1	Temperature Setpoint	CRAC-1	N/A	VND-162, N-2-2, NAE-05
A-2	Temperature Sensor	CRAC-1	N/A	MIG-008, NAE-14, Trunk-1
A-2	Zone Temperature Sensor	VAV-327	Floor-3, Room-W-Hall	Trunk-2, NAE-13, N-2
B-1	Zone Temperature Sensor	N/A	Room-123A	N/A

An entity is an instance of a TagSet in Brick and a TagSet is composed of Tags. *Temperature Sensor* is a TagSet with the Tags *Temperature* and *Sensor*. Brick introduces a class hierarchy to organize the TagSets. At the highest level, they have *Location*, *Equipment* and *Point*. *Equipment* is sub-classed into Light, Fan, etc. *Points* are timeseries data streams such as *Sensors* and *Setpoints*. *Location* can be *Room*, *Floor*. Fig. 2 depicts a part of the Brick hierarchy.

Fig. 3 compares the raw metadata of an example point and the equivalent Brick representation of it. It represents a 'zone temperature sensor' in 'room 3.' A zone refers to the area affected by an air conditioning unit. The Brick representation explicitly describes that ZNT is an instance of *Zone Temperature Sensor*, RM3 is an instance of *Room* and ZNT is located in RM3 using Turtle syntax [7].

2.3 Related Work

Various approaches have been proposed for reducing the effort to create an instance of such metadata schema from diverse data sources. Such methods span over all the aspects of machine learning such as active learning [6, 9, 14], supervised learning [12] and transfer learning [13]. The primary goal of all the work is to learn useful semantic mapping from given information such as timeseries data and raw metadata to usable labels with the least human effort. However, each of the work has different focus. Zodiac [6] is an active learning framework modeling the raw metadata as Bag of Words (BoW). BoW ignores the order of the words so it can model different types of metadata such as BACnet unit and vendor-given name in the same way, with which Zodiac achieves high precision inference. However, it only focuses on point types which makes the model simple. Bhattacharya et al. [9] propose a framework learning synthesis rules for vendor-given names from examples. It can effectively learn the parsing rules covering all the patterns inside the target building. However, it has several shortcomings to be generalized for practical usage. First, it cannot parse a long metadata as the number of the necessary rule combinations increases exponentially along with the length of each metadata point, and may take a long while to deal with a corpus of points with even 80 characters. It is typical to have 60-80 characters in modern BMSes while some old buildings only have 15-20 characters. Second, it only parses a sequence of characters while there can be multiple sequences describing a point. The information in different columns of metadata can identify the contexts of the point more clearly and richly. For example, Degree Celsius in BACNet unit may help identifying a point is associated with temperature. Third, it is not capable of reusing existing mappings even though the buildings may have similar patterns. Scrabble takes a hybrid approach with two stages, modeling both sequences and BoW in a systematic way to achieve trasferrable mappings that exploit different types of metadata. There are other metadata inference mechanisms heavily using timeseries data such as BuildingAdapter [13] for transfer learning and Gao et al. [12], but we focus on understanding the raw metadata as they are commonly accessible, have much richer information and more stable than learning from timeseries data. As detailed in Section 4.3, Scrabble can infer any kind of entity in raw metadata unlike Zodiac and achieves higher accuracy than Bhattacharya et al.'s algorithm in addition to the capability of reusing existing buildings' mapping for a new target building.

Algorithm 1 Scrabble Process Overview.
B_S: source building, B_T: target building. s_i:

```
procedure SCRABBLE($B_S, B_T$)
    while $U(s_i) < th \; \forall s_i \in B_T$ do
        Use CRF to learn Characters from $B_S \cup D$.
        Use MLP to map Tags → TagSets from $B_S$
        Infer Tags of points in $B_t$ with the CRF model.
        Infer TagSets for the Tags of points in $B_T$ with the MLP.
        Select samples with low confidence or utilization.
        Resolve manually on chosen samples.
    end while
end procedure
```

Project Haystack [3] is another popular standard metadata schema. Instead of Brick's TagSet/class concept, a user associates only tags to an entity in Haystack. Tagging schema is structurally a subset of TagSet as TagSets consists of Tags. Thus, Haystack is a subset of Brick in terms of expressivity so algorithms that can infer Brick can easily do for Haystack. In Scrabble, the first stage using CRF alone is sufficient to infer Haystack as it maps raw metadata to Brick Tags, which is theoretically equivalent to Haystack tags.

3 SCRABBLE

Given unstructured metadata for data points in a target building B_T and ground truth semantic labels for points in a source building B_S, we normalize the metadata of the target building B_T into the

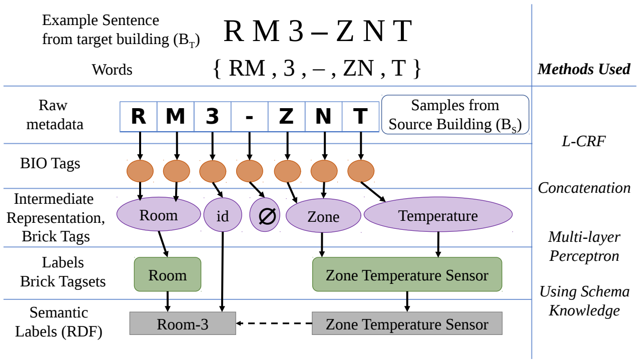


Figure 4: Data mapping from raw metadata to semantic metadata. CRF maps tokens in raw metadata to the intermediate representation, Brick Tags. Multi-label classifier maps Tags to final labels, TagSets.

structured Brick schema. Our goal is to minimize the number of examples to extract correct and comprehensive metadata present in the given unstructured metadata.

Scrabble uses an active learning framework for normalizing metadata of sensors in multiple buildings by adopting a transferrable intermediate layer. We map a sentence from the target building to a set of Brick Tags using a CRF classifier that trains on samples from source buildings and examples from domain experts. We use Brick Tags as a reusable Intermediate Representation (IR) that is free of building specific metadata. IR is effective in reducing the number of learning samples if it is easier to learn mapping from inputs to IR than labels directly [11, 17, 21]. A multi-layer perceptron (MLP) then maps the Tags to corresponding TagSets. MLP is capable handling high dimension data very well by learning important features inside hidden layers. We use confidence-based and a Tags utilization metrics to identify samples likely to be labeled incorrectly and resolve them by input from domain experts. Scrabble iterates through all target building samples until it can identify all building labels with high confidence. Algorithm 1 summarizes the entire process and Figure 4 describes the mapping of raw metadata to the semantic metadata with an example.

Our technique is based on two key observations. First, a word's meaning does not change even if its usage varies in different sensors. In Figure 4, RM is used to indicate *Room* as a sensor's location whereas it can be a part of RMT to represent *Room Temperature* in another sensor. The meaning of RM as a *Room* remains identical in both cases. Thus, mapping from raw strings to Tags can be reused across different buildings though target labels may differ. Second, the relationships between Tags and TagSets can be learned and used across different buildings. For example, across all buildings, the Tags *Temperature* and *Sensor* form the Tagset *Temperature Sensor*. Thus, Tags can be reusable representations of metadata across buildings, which can in turn be easily mapped to Brick schema. We adopt this idea from zero-shot learning methods that use semantic codes [17] and attributes [21]. We reuse the relationships discovered in a source building when available and domain experts can provide samples to learn newly observed relationships. We can rapidly retrieve structured metadata from a new building in this way.

3.1 Terminologies

Figure 4 gives an example for the terminologies. A building has various *points* that produce a data stream such as sensors. BMSes describe points with various types of *raw metadata*.Raw metadata associated with a point represents a set of labels, for which we use Brick *TagSets*. Raw metadata may contain *string metadata* like point names and *code metadata* like units. We call string metadata also a *sentence* composed of multiple *words*. A word may represent a set of Brick *Tags*, such as RM represents `Room`, but it does not have to be delimited by special characters. A word is decomposed to *characters*, of which each is mapped to a *BIO token* [20] associated with its word's Brick Tags (details in Section 3.2.) In the learning process, a human *expert*, such as a building manager, provides *examples* for mapping a sentence to 1) Tags and 2) the TagSets that the point's raw metadata represent. Required examples are chosen by Scrabble to minimize the total amount of effort.

3.2 Raw Metadata to the Intermediate Representation

We define a word as a set of characters representing a concept. E.g., ZN, T, and RM in Figure 4. We map words to Brick Tags as an intermediate representation. Here, words are not necessarily separated by predefined delimiters. For example, ZN and T in ZNT are separable as *Zone* and *Temperature* because they can be reused in other sentences such as RMT for *Room Temperature*. In contrast, mapping ZNT directly to *Zone Temperature* loses the reusability of ZN and T in other contexts.

Every character in a word is labeled with a BIO (Begin, In, and Out) token [20] to represent the location in the word. The word ZN corresponds to the Tag *Zone*, of which Z is located at the Beginning of ZN and N is Inside ZN. The BIO scheme captures this relative position of the character in the word and assigns Z to "B-Zone" and N to "I-Zone". The "O" BIO tag stands for Out, i.e., tokens that do not convey semantic meaning such as punctuations and definitive articles. In Figure 4, the punctuations ',' and '-' will be assigned to "O" token.

Scrabble learns a Conditional Random Fields (CRF) model [15] for mapping raw building metadata to Brick Tags with BIO tokens, e.g it learns that ZNT corresponds to "B-Zone", "I-Zone" and "B-Temperature" with examples from source building. CRF makes a Markov independence assumption, i.e., the tag of a character only depends on the neighboring characters. We use the following as input to our CRF model for character j: the jth character itself, (j-1)th character, (j-2)th character, (j+1)th character, is digit? and is special character?

We additionally adopt code-based metadata such as BACnet units other than textual metadata. BACnet defines codes for certain entries such as units and object types [10]. For example, unit code 62 represents Celsius in BACnet, with which we surely know that the point is associated to *Temperature*. In a similar way, a point with object type "analog input" in BACnet can be considered as a *Sensor*. These metadata are scattered in different entries other than in a single string so methods only parsing a string cannot integrate them systematically. Scrabble merges Brick Tags from different metadata at the second stage. Though we only use BACnet metadata, which is common in BMSes, this concept can be generalized into any other code-based metadata that can be interpreted as Brick Tags.

3.3 Mapping Intermediate Representation to Semantic Labels

We have thus far mapped the raw building metadata to Tags, which are an intermediate representation (IR) of Brick TagSets, our target semantic label. Our seconds stage maps Tags to TagSets. As the Brick schema can represent a general building, a classifier that labels TagSets from Tags can be shared across different buildings. Mapping from Tags to TagSets is challenging because not all Tags of a TagSet may be present when we perform the raw metadata to IR mapping, e.g., *Sensor* is often omitted in *Zone Temperature Sensor*. A single Tag can be attributed to different TagSets, and we need to identify which of these TagSets is the correct semantic mapping. For example, the Tag *Room* may correspond to the location TagSet *Room* or to a point TagSet like *Room Temperature Sensor*.

We use Bag of Words (BoW) with Term-Frequency Inverse-Document-Frequency (TF-IDF) [22] scheme to vectorize IR. BoW counts the occurrence of each Tag learned from a sentence and stores it as a feature vector. The length of the feature vector, i.e. its dimension, is the number of Tags in the Brick schema. TF-IDF skews the Tag counts to reduce the importance of common words such as 'the', 'to'. Learning standard classifiers on BoW such as a decision tree is an intuitive approach to learn the mapping the between IR and TagSets. However, there are several difficulties to learn such classifiers. First, a set of Tags generated from a sentence represents multiple TagSets, which is a multi-label classification problem. In our model example, the raw metadata "RM3-ZNT" describes a *Zone Temperature Sensor* and a *Room* simultaneously (Figure 4). As the relationships among Tags for a sensor are unknown, what Tags in the set are used for what types of TagSets is also unknown. We need to identify multiple labels from one distribution. From the set of Tags, {Room, id, ∅, Zone, Temperature}, in Fig. 4, we need to identify two TagSets, {Room, Zone Temperature Sensor}. Second, samples are significantly biased across different buildings. There are points widely used such as *Zone Temperature Sensor* while some points specific to control special equipment occur only once. In the four buildings of our data set, 17% of TagSets account for 90% of TagSet occurrence on average and 34% of TagSets occur only once. Moreover, different buildings would have different types of points and metadata. A specific type of equipment may exist only in a particular building. Such new information cannot be pretrained from a source building. To address these challenges, we propose three approaches in Scrabble: sample augmentation, a domain agnostic classifier and iterative sample selections.

3.3.1 Sample Augmentation. Samples in a building are inherently biased toward the building's configuration and its original installer's writing style. We synthetically generate samples from the schema and the given building's samples to mitigate the biased samples in three ways.

(a) Brick Samples: The schema provides samples of mapping from Tags to TagSets as a TagSet is a composition of some Tags (e.g., *Zone*, *Temperature* and *Sensor* for *Zone Temperature Sensor*). We insert sets of Tags for each TagSet and a Tag in each set has a random frequency from 1 to a threshold, th_{tag}, where th_{tag} is the median frequency of Tags inferred in a sensor of a source building. The number of the generated schema samples, th_{ts}, is a hyper-parameter chosen empirically with a validation set. We limit each schema sample to have just one label. For instance, we generate a sample mapping tags of (*Temperature*, *Sensor*) to a TagSet, (*Temperature Sensor*). We avoid adding samples with multi-labels because the number of possible combinations of different TagSets increases exponentially to the number of total TagSets.

(b) Negative Samples: We can also add more possible combinations of TagSets from the given samples using a logic similar to Brick Samples. Given a set of labels for a sample, we add its variations without a TagSet by removing Tags related to the TagSet (e.g., remove the Tag *Room*} to remove the TagSet *Room* in the label set of *Room*, *Zone Temperature Sensor*}.) It prevents overfitting to given samples while generating tof an acceptable number of samples in the order of the number of given samples.

3.3.2 Multi-layer Perceptron for Multi-label Classification. We use Multi-layer perceptron (MLP) to model the mapping from Tags to TagSets. MLP uses fully connected (FC) neural network layers to act as a universal function approximator [19]. Here, we use sigmoid as the non-linearity function and use binary cross-entropy for the loss evaluation in the training phase. We use 2 FC layers and each of them is followed by a dropout [25] layer to generalize the model. This MLP stage receives vectorized BoWs for Brick Tags as inputs and infers a set of TagSets the vector represents as outputs. Our input data are considerably hard because the Tags are in high dimensions but sparse. We have empirically evaluated other multi-label classification models such as classifier chains, random forests, but MLP outperforms other methods because of its dimensionality reduction ability.

3.4 Sample Selection

We cluster sentences based on the tokens to identify similar sentences [6, 9]. Each sentence is converted to a vector of the BoW model with tokens usually defined by contiguous alphabets or special characters. This tokenization need not be precise as it just needs to recognize similarities among sentences. The vectors are clustered based on hierarchical clustering and a small threshold determines output clusters. Balaji et al. [6] empirically show that if the threshold is small enough, sentences in a cluster have the same label for sensor type, which is the most complex information in metadata. When selecting an example to learn, we pick one randomly from the most uncovered cluster. The coverage is defined as the rate of sensors given examples over the number of sensors in a cluster. This method is generally applicable for determining what examples an expert should provide to derive the best learning speed.

3.5 Active Learning with Domain Experts

We iteratively update the learned model with the target building's sample given by an expert. Target building's raw metadata may contain some points unobserved from the source buildings such as new systems and new conventions that should be taught by an expert. We have to carefully select samples for experts to answer so that we can achieve the fast learning ratio with minimal examples, which is called Active Learning (AL). In general AL, we evaluate unlabeled samples with the learnt model and pick the most informative samples based on a certain query strategy. A domain expert provides labels for the samples, we learn a new model with the added samples, and iterate the entire process.

As Scrabble consists of two-stages with one from characters to Tags with CRF and the other from Tags to TagSets with MLP, a query strategy should pick good examples covering both stages. We exploit two types of query strategies for different stages. The first one is to find the lowest confident inference at the CRF stage and the lowest entropy of inferences at the second stage. CRF is a sequence model that maps a sequence to labels. Among various ways of querying strategies [24], we select least confident inferences (LC) for its simplicity, interpretability and less computational complexity. Settles and Craven [24] show that LC's performance is competitive compared to the others with less computational cost. Confidence of a CRF inference is an inferred sequence's conditional probability but we normalize it with its length because the probability highly depends on the length of the sequence.

$$\Phi(\boldsymbol{s}, \boldsymbol{b}, \theta) = log(P_\theta(\boldsymbol{s}, \boldsymbol{b}))/length(\boldsymbol{s}), \tag{1}$$

where $\boldsymbol{s}$ is a sentence with characters, $\boldsymbol{b}$ is BIO tags inferred from the model θ. P_θ is a CRF's loss function[1].

The second stage is multi-label classification and single label confidence cannot represent how the inference is confident in general. We instead exploit an assumption that identified Tags should be fully mapped to certain TagSets and query how much Tags are exploited in the current inferences. The assumption is based on the observation that the original installers put only meaningful information into the raw metadata as the space for the metadata is limited and it needs to provide useful information for maintenance and operations. For example, if given Tags are Temperature and Sensor, mapping it into Sensor is incomplete as Temperature is not used in the inference. Tags utilization is defined as follows:

$$U_{tags}(\Theta_i, T_i) = \frac{\sum_j usage_{tags}(\theta_{i,j}, T_i)}{length(s_i) - \#(O\text{-}Tag)} \tag{2}$$

$$usage_{tags}(\theta_{i,j}, T_i) = \begin{cases} 1, & \text{if } \exists t_{i,m} \mid \theta_{i,j} \in t_{i,m} \wedge t_{i,m} \in T_i \\ 0, & \text{otherwise,} \end{cases}$$

where Θ_i is a set of Tags identified for the point and T_i is a set of TagSets inferred from Θ_i. $\theta_{i,j}$ are Tags in Θ_i and $t_{i,m}$ are TagSets in T_i. O-Tags are ignored in the calculation as they have little meaning. We pick target building samples with low utilizations by the diverse random sample selection method. We dynamically choose outliers with utilizations less than an average of the entire utilizations subtracted by their standard deviation.

The entire process is summarized in Algorithm 1. Models for both Tags (intermediate representation) and TagSets (labels) are initially learned from a source building. We infer Tags and TagSets in the target building. Then, we calculate all utilizations of metadata at the target building and randomly choose N diverse samples. We set N as 0.5% of the target dataset size, but it is a hyper-parameter dependent on heterogeneity of a target data set and the learning speed of a user's preference. An expert provides labels of the asked samples such as positions of words, their corresponding Tags and TagSets. All the models then are learned with the updated data set. These steps are iterated until all the utilizations of raw metadata in the target building is higher than a threshold.

4 EXPERIMENTAL EVALUATION

4.1 Experimental Setup

4.1.1 Datasets and buildings. We evaluate the framework with three buildings from campus A (A-1,2,3), one from campus B (B-1) and the other one from campus C (C-1). C-1 is one of the buildings used in ProgSyn [9]. Due to the huge amount human effort for labeling, we randomly choose 1000 examples per building for our evaluation in A-1,2,3 and B-1, and verify them manually for ground truth. However, we use the entire points in C-1 to exactly compare Scrabble's performance with ProgSyn. Table 2 summarizes the statistics of the buildings. Missing Tags indicates the number of Tags shown by its corresponding TagSets. With less missing Tags, it would be more straightforward to map them to the target TagSets. A-1,2,3 and B-1 have more informative metadata with a longer metadata size and the contained TagSets. C-1 has relatively concise metadata with 21.7 characters metadata in average.

We first analyze the difference between the buildings for raw metadata and ground truth labels. Fig. 5a describes the rates of words in B_T occurring in B_S to show the possibility of reusing the model from source buildings. Words are defined as contiguous letters with a unit meaning that can be mapped to Brick Tags. Words coverages equally weight each word and the weighted word coverages includes frequencies of words in each building. It shows that at least a half of the words in B_T can be obtained given B_S, but the other half has to be learned with expert input. The gap between word and weighted word explains more frequent words are more common across different buildings. Thus, using B_S would help understand common words in B_T. These similarities are directly reflected in the learning rates of Scrabble in the later sections.

There are 119 TagSets and 101 Tags in a building on average among the 904 TagSets and 284 Tags defined in Brick. The similarities of Tags and Tagsets across buildings are shown in Fig. 5b. Tags model would be more transferrable between buildings as Tags are more shared between buildings than TagSets. We suspect Scrabble will perform better when the source and target buildings have similar metadata style. However, even when target building metadata style is different, our hypothesis is that the learning rate would be better compared to learning from scratch.

4.1.2 Implementation. We implement Scrabble[2] in Python with PyCRFsuite [16] for CRF, Keras [2] for MLP and scikit-learn [18] for the other machine learning algorithms. All the datasets are stored as files and the ground truth datasets are used as a domain expert iteratively providing labels for active learning.

4.2 Evaluation Metric

We infer a set of TagSets from raw metadata. We use a label-based and an example-based metric [26].

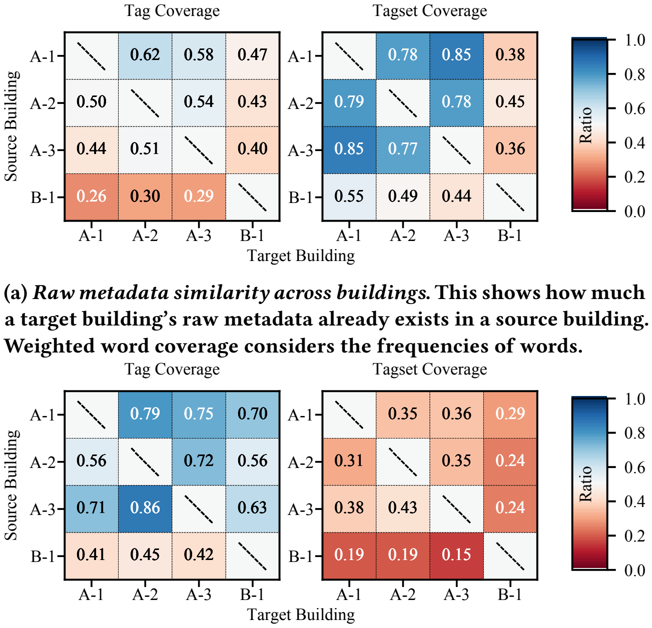
- $Accuracy(h) = \dfrac{1}{p}\sum_{i=1}^{p} \dfrac{|Y_i \cap h(x_i)|}{|Y_i \cup h(x_i)|}$,
- $MacroF_1(h) = \frac{1}{q}\sum_{j=1}^{q} F_{1,j}$

[1] For the exact CRF loss function, please refer the original literature [15].

[2] Scrabble repository: https://github.com/jbkoh/scrabble

Building	Points (selected)	Total TagSets	Total Tags	Avg TagSets	Avg Required Tags	Avg Existing Tags	Avg Missing Tags	Avg Len of Metadata
A-1	4593 (1000)	129	122	6.30	8.67	7.83	1.09	67.7
A-2	1914 (1000)	120	96	7.12	9.23	8.62	0.83	67.9
A-3	4381 (1000)	137	111	6.90	9.85	9.00	0.94	74.9
B-1	8363 (1000)	90	73	4.37	6.36	6.30	1.35	59.4
C-1	2551 (2551)	68	65	3.03	5.17	3.69	2.71	21.7

Table 2: Quantities of datasets. We choose 1000 points from each building randomly for the evaluation. The numbers of Tags and TagSets explain diversity of labels in buildings. The numbers are also averaged over data points. 1.1 Tags should be learned from example patterns, which accounts for 26% of a TagSet in average.



(a) *Raw metadata similarity across buildings.* This shows how much a target building's raw metadata already exists in a source building. Weighted word coverage considers the frequencies of words.

(b) *Tags/TagSets coverage:* Coverage is the rate of common Tags and Tagsets between two buildings over those in the target building, showing Tags are more common than TagSets.

Figure 5: Similarity Comparison across Buildings' Datasets

where h is a model, p is the number of samples, q is the number of labels, x_i is an ith input vector and Y_i is ith label set. $h(x_i)$ produces x_i's inferred labels. *Accuracy* calculates an average ratio between the number of correctly inferred labels and the sum of the correct labels, irrelevant labels and misclassified labels per sample. It captures how well the entire data set is classified, but the metric can be skewed by dominating classes. In contrast, Macro-averaged F_1 (Macro F_1) captures the normalized mean of measures F_1 scores across classes. It can exaggerate incorrect inferences of classes with rare samples though it provides a good estimation of the model's class coverage. In our dataset, a few classes such as the name of the *Building* occurs in all points, while a few classes such as pump occur in very few. Such disparity in class samples causes the two metrics to differ significantly.

4.3 Baselines

For our baselines, we use two algorithms; the program synthesis (ProgSyn) [9] and a multi-label version of Zodiac [6]. ProgSyn is a promising solution for parsing strings but has several critical drawbacks as discussed in Section 2.3. In addition to the drawbacks, the algorithm cannot parse a string if there are repeated labels. In A-1,2,3, there can be repeating labels across metadata types, so ProgSyn simply cannot be executed over the datasets. Thus, ProgSyn is excluded in buildings other than C-1 in our evaluation.

Due to the limitation of the ProgSyn, we devise another baseline. Zodiac is an active learning framework for inferring point types from raw metadata while Scrabble extracts all possible labels - point type, equipment type, location. Zodiac vectorizes the raw metadata with BoW scheme and learn a multi-class classifier but it is limited to infer point types only. We modify Zodiac to infer multiple labels. We first use TF-IDF scheme instead of count vectorization to account for variation in word frequencies across different buildings. We use Classifier Chain of RF Classifiers for multilabel classification instead of a single RF classifier. Zodiac uses confidences of predictions to determine the most uncertain samples to ask to experts, but there is no single confidence representing the inference on a sample in multi-label classification because each class has its own confidence. We instead use entropy of class probabilities per sample [24]. The rationale is that if a sample is confidently inferred, its confidences of each class will be close to either zero or one. Entropy is high when a distribution is extreme and we can know that if a multi-label inference is not confident when an entropy of the inferences is low. An expert is asked to provide labels for 10 samples with the lowest entropy for each iteration. Note that mapping a BoW vector to a set of TagSets has a limitation of losing the relationship between actual words and the labels. For example, it cannot associate a *Room* label to the portion in the string representing the room including its identifier like room numbers. It is very often important to identify actual room number or device IDs for applications to properly take actions for the target object. Scrabble can traceback the mappings from characters to actual TagSets and identify actual names together.

Baseline results of five buildings are shown in Fig. 8. For C-1, ProgSyn's learning rate is steep in the early stage, but it slows down soon. Furthermore, it cannot reach 99% *Accuracy* because it lacks in the capability of accumulating more examples once it reaches the point where every metadata are qualified. For multi-label Zodiac, we observe that the words are similar in the buildings in the same campus A as their accuracies are initially higher than one being transferred from/to B-1. In the baselines, accuracies initially increase rapidly with the sample numbers because samples are biased. A few examples can represent many others. Macro F_1 increases linearly which shows that the sample query mechanism is valid.

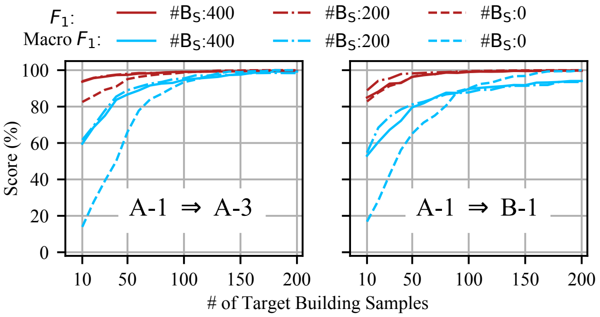


Figure 6: *Learning rate of CRF mapping characters to Brick Tags.* It compares the learning rate with the different numbers of source building examples.

In all cases, the baseline has no notable gain with source buildings initially with 10 examples. It shows that it is hard to exploit different buildings together with the naïve features. All of the buildings have the problem in achieving high *Accuracy* like 99% within 200 examples. For example, its best case is achieves 99% accuracy at 210 examples in A-1 and the worst case is 98% accuracy at 300 examples in A-3. We conclude that we need a framework converging faster.

4.4 Experimental Results

We individually evaluate the first stage (characters ⇒ Tags), the second stage (Tags ⇒ TagSets), and the entire framework. Our experiments initially take 10 examples randomly chosen by the method in Section 3.4 and 10 examples per iteration based on the methods in Section 3.5 for 20 rounds. All results are averaged over four experiments.

4.4.1 CRF Evaluation. Scrabble maps characters to BIO token labels with CRF and concatenates them to form Brick Tags. Fig. 6 shows the results of learning the tags of a target building with source building data. Initially, 10 samples are randomly selected based on the mechanism in Section 3.4 and samples of the source building are uniformly randomly chosen. In each iteration, the models are learned and tested, and then the expert gives 10 different examples with lowest confidences. The metrics in the figure are for Brick Tags, which are the features used in the next stage.

Fig. 6 shows that 200 samples from B_S improves F_1 from 82.6% to 93.7% and $MacroF_1$ from 14.0% to 61.7% in A-1⇒A-3, and from 82.8% to 88.9% and from 16.8% to 54.7% in B-1⇒A-1 initially than learning without B_S samples. A-3 easily benefits from A-1 as they are in the same campus with similar conventions as described in Fig. 5a. F_1 is high as 94% from the beginning with A-1. Even without A-1, F_1 is high as 82.0% because of several dominant words such as building names or location like Room occurring in most of the raw metadata. MacroF_1 also improves from 14.0% to 61.7 % and 60.0% by adding 200 and 400 examples from A-1, exploiting the similarity between A-1 and A-3. Even though the sample query mechanism is valid as the metrics monotonically increase, MacroF_1' converge around 100 samples. There is little difference between 400 samples and 200 samples for A-1 ⇒ A-3.

However, for buildings from different campuses (A-1 ⇒ B-1), we observe that MacroF_1 converges lower when there are source data than one without source data. Still, there is a large gain in MacroF_1 initially due to certain similarity of common words such as Room. The degradation of $MacroF_1$ in the later stage would be due to the overfitting. Hyperparameter optimization needs to be investigated more. In general, adding samples from B_S improves learning rate in all cases initially, but may decrease $MacroF_1$ when there are many examples with different styles. It shows Scrabble's initial transferrability but more generalizability with large data sets should be investigated more.

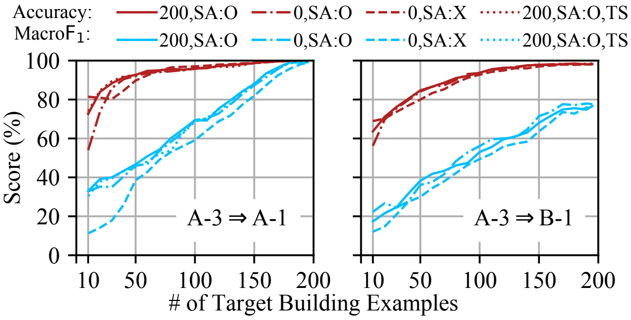


Figure 7: *Learning rate comparison of different configurations for TagSet classifier* based on ground truth Tags. SA stands for sample augmentation, TS means learning with timeseries data, and the numbers are the number of samples from a source building.

4.4.2 TagSet Classifier. To independently evaluate the performance of the TagSet Classifier, we presume the TagSet classifier receives correct Tags from the first stage. Fig. 7 shows how each component improves the learning rate of a target building based on *Accuracy* and Macro F_1. The naïve scenario without sample augmentation and samples from the source building achieves low $MacroF_1$ as 11.1% and moderate 81.2% *Accuracy* initially. Sample augmentation (SA) in the experiments includes all the two types discussed in Section 3.3.1. As it provides valid examples not included in the given small B_T's data set, $MacroF_1$ is also increased from 11.1% to 36.5% though the accuracy rather decreases. In the naive scenario, common TagSets such as Room can be simply identified by a few examples, represented by initial high accuracy but low MacroF_1. While the benefit of this layer is consistently higher than naïve approach between similar buildings, it is unclear for buildings in different campuses.

4.4.3 Overall Performance. We evaluate five directions of active learning with knowledge transfer to cover different types of coverages found in Fig. 5a. While multi-label Zodiac and Scrabble select 10 examples per iteration, ProgSyn picks a sample in each iteration from the beginning. Due to the long experiment time as five minutes to a few hours per iteration in learning CRF model, we restrict samples per source building to 200 instead of the entire 400. The analysis in Section 4.4.1 already shows that learning with 200 B_S samples gives similar results to 400 samples for the first stage.

As an active learning framework, Scrabble outperforms both ProgSyn and multi-label Zodiac for *Accuracy* and $MacroF_1$ in general. In C-1, Scrabble has 83.2% *Accuracy* and 25.3% $MacroF_1$ while ProgSyn shows 42% and 12% initially with 10 samples. Scrabble also continuously outperforms and can reach 99% *Accuracy* while ProgSyn cannot. Compared to multi-label Zodiac, *Accuracy* and

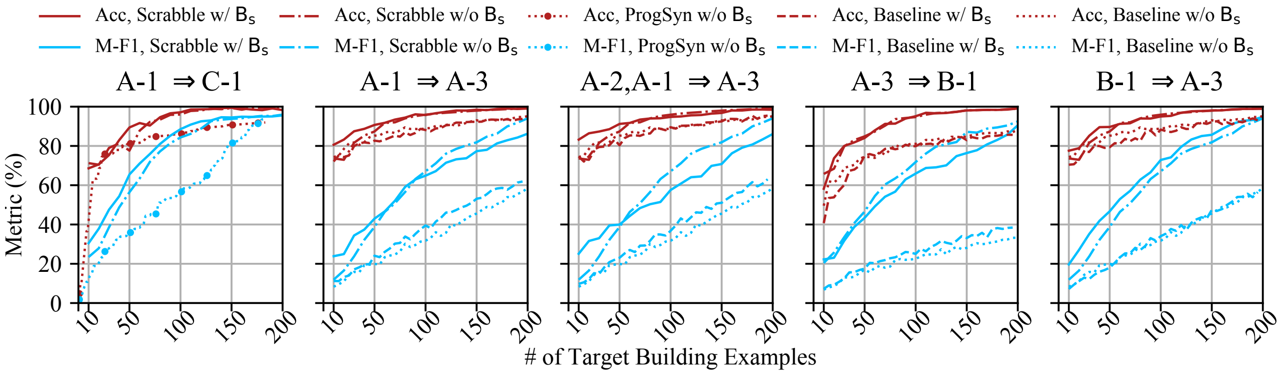


Figure 8: *Learning Rate of Scrabble's Entire Process.* At the left side of each arrow are source buildings and at the other side is a target building. Note that ProgSyn is only working with C-1 due to its limitations. ProgSyn also ceases before achieving high accuracy because of its deterministic algorithm. In general, Scrabble shows better performance exploiting existing buildings in the same campus with similar patterns. Even across different campuses, Scrabble does not degrade accuracies.

$MacroF_1$ of the best case of A-1,A-2 ⇒ A-3 are 83.2% and 25.3% initially with 10 samples, while multi-label Zodiac's are 74.6% and 9.78%.

Furthermore, it shows the knowledge accumulation by showing adding more examples from source buildings results in similar or better initial inferences. *Accuracy* and $MacroF_1$ of learning A-3 without source buildings are 73.6% and 11.9%, which 8.6% and 15.5% lower than learning with source buildings initially. However, we again experience $MacroF_1$ degradation in later stages possibly caused by overfitting. Our hypothesis is that source buildings' data distribution could confuse the evaluation metrics used for sample selection. The coverage of the words and Tags would affect the results as well. While using B-1 for A-3 improves the accuracies consistently, using A-3 for B-1 disturbs the learning rate. A-3 has more diverse patterns than B-1 as in Section 4.1.1, which would add noise to B-1's model more than the other way.

Overall, Scrabble always shows better performance than the other baselines including ProgSyn and the multi-label Zodiac. When 10 samples are initially given, Scrabble has 67%/31% *Accuracy*/$MacroF_1$ than ProgSyn's 42%/12% for C-1, and 83%/25% than the multi-label Zodiac's 75%/9.8% at best. To achieve 99% *Accuracy*, Scrabble requires 100 examples for C-1 while ProgSyn cannot achieve. For A-3 using A-1 and A-2, Scrabble requires 160 examples while the modified Zodiac needs 280 examples for 98 % *Accuracy*. Scrabble shows the possibility of reusing the models with intermediate representations though a model that is more generalizable across different data patterns is still a future research topic.

5 DISCUSSION

5.1 Learning from Time-Series Features

A way to augment the transfer learning process is by using time-series data, of which features are more common across different types of buildings [13]. Tags can be differentiated using the features from the data collected by the building [12]. E.g., a *Temperature* related sensor may have an average around 70^oF for indoor temperature. We choose the source building to be A-1 and the target building to be A-3. We extract 16 time-series features, such as mean and Fourier Transform values, from both buildings to train a random forest model as a multilabel classifier to model Tags that points represent in the source building. We infer Tags of interest at the target building with the learned classifier. The result is shown in Fig. 9, where we compare the precision and recall.Fig. 9 shows that validation and test sets track each other, which indicates that time-series features can be used to augment the transfer learning process. However, the competence to augment the transfer learning process is limited to how diverse the data is at the source building. From Fig. 5b we can see that the Tag coverage for A-1 ⇒ A-3 is 75%, this means that our model, initially, will not be able to represent 25% of the Tags for building A-3. Furthermore, Tags occurrences in buildings are significantly biased as discussed, there are quite a few Tags which only have time-series data representing those particular Tag, which is not enough data to train the model adequately. Only several Tags with significant numbers can be properly modeled by timeseries features.Hence, it limits the effectiveness of augmenting the learning process. We add the Tags determined by timeseries features to the ground truth and test the TagSet classifier's performance as shown in Fig. 7. Its accuracy is similar to the the original TagSet classifier, but Macro F_1 decreases due to the improper representation of rare Tags by timeseries features.

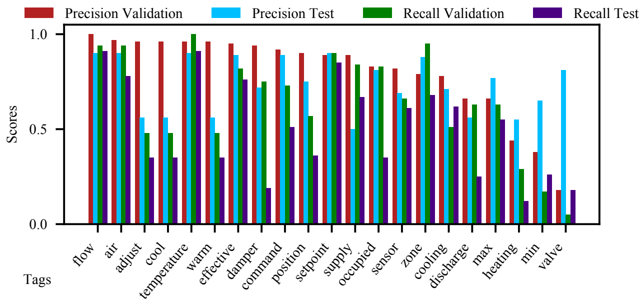


Figure 9: *Tags Inference from timeseries features* with source building (A-1) at the target building (A-3).

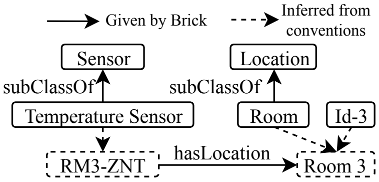


Figure 10: ***An example of semantic postprocessing with Brick.***

5.2 Semantics Postprocessing

We identified TagSets from raw metadata. There are more information embedded in the raw metadata both explicitly and implicitly. We infer identifiers (IDs) of entities in the same way as TagSets though they are not included in the evaluation metrics because they may overstate our performance. Two types of IDs such as equipment number or a hall name are positionally defined to indicate whether they qualify a TagSet right or left to the ID. We can glue the IDs to TagSets to reconstruct original entities' name. In Fig. 10, ID's position confines *Room* to Room 3. *Temperature Sensor* does not have corresponding ID, but rather it is common to use the entire name as an ID. Another type of information is relationships between TagSets. A *Temperature Sensor* may reside in a *Room*, which is not explicit in metadata but a domain expert can infer. Such relationships can be also inferred by a schema with relationships such as Brick defines canonical relationships between entities in buildings. In Fig. 10, we can explicitly know that "RM3-ZNT" is a *Sensor* and "Room 3" is a type of *Location*. The only possible relationships in Brick for them is *hasLocation* so we can explicit the semantic relationships among them without human intervention. The above two mechanisms are excluded in our analysis due to the lack of technical novelty but notable for practical use.

6 CONCLUSION

We have proposed and evaluated Scrabble, a framework to normalize unstructured metadata by exploiting the schema mapping from known buildings to a target building. Scrabble uses a schema, Brick, for an intermediate representation of labels, as well as, the labels that raw metadata may represent. We have shown that Scrabble outperforms 59%/162% in terms of Accuracy and Macro-Averaged F_1 than the prior work while allowing it to achieve 99% Accuracy with 100-160 examples for buildings with thousands of points. While various building schemata have been in the interest of smart building researchers and field engineers for building application deployments, instantiating such schemata has required huge effort, blocking adoption of modern applications by its cost. Scrabble will help adopting such schemata for building applications with their potential to save energy and improve the quality of life for people.

ACKNOWLEDGMENTS

Our sincere thanks to National Science Foundation for supporting this work [CNS-1526841, CSR-1526237, TWC-1564009, and IIS-1636879] and gift from Johnson Controls, supporting Brick. We also appreciate our colleagues, Arka Bhattacharya, Lawrence Saul, Ndapa Nakashole, for the productive discussions.

REFERENCES

[1] [n. d.]. Apple HomeKit. https://www.apple.com/ios/home/.

[2] [n. d.]. Keras. https://keras.io/.

[3] [n. d.]. Project Haystack. http://project-haystack.org/.

[4] U.S. Energy Information Administration. 2016. User's Guide to the 2012 CBECS Public Use Microdata File. *Commercial Buildings Energy Consumption Survey (CBECS)* (May 2016), 33.

[5] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. 2016. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the ACM International Conference on Embedded Systems for Energy-Efficient Built Environments (BuildSys). ACM.*

[6] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. 2015. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments.* ACM, 13–22.

[7] David Beckett, Tim Berners Lee, Eric Prud'hommeaux, Gavin Carothers, and Lex Machina. 2014. RDF 1.1 Turtle. Retrieved May 15, 2017 from "https://www.w3.org/TR/turtle/".

[8] Arka Bhattacharya, Joern Ploennigs, and David Culler. 2015. 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.* ACM, 33–34.

[9] Arka A Bhattacharya, Dezhi Hong, David Culler, Jorge Ortiz, Kamin Whitehouse, and Eugene Wu. 2015. Automated metadata construction to support portable building applications. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments.* ACM, 3–12.

[10] Steven T Bushby. 1997. BACnetTM: a standard communication infrastructure for intelligent buildings. *Automation in Construction* 6, 5-6 (1997), 529–540.

[11] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. 2009. Describing objects by their attributes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 1778–1785.

[12] Jingkun Gao, Joern Ploennigs, and Mario Berges. 2015. A data-driven meta-data inference framework for building automation systems. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments.* ACM, 23–32.

[13] Dezhi Hong, Hongning Wang, Jorge Ortiz, and Kamin Whitehouse. 2015. The building adapter: Towards quickly applying building analytics at scale. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments.* ACM, 123–132.

[14] Dezhi Hong, Hongning Wang, and Kamin Whitehouse. 2015. Clustering-based active learning on sensor type classification in buildings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management.* ACM, 363–372.

[15] John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, Vol. 1. 282–289.

[16] Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). http://www.chokkan.org/software/crfsuite/

[17] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems.* 1410–1418.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[19] Allan Pinkus. 1999. Approximation theory of the MLP model in neural networks. *Acta numerica* 8 (1999), 143–195.

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

[21] Bernardino Romera-Paredes and Philip HS Torr. 2015. An Embarrassingly Simple Approach to Zero-Shot Learning.. In *ICML.* 2152–2161.

[22] Gerard Salton and Michael J McGill. 1986. Introduction to modern information retrieval. (1986).

[23] Anika Schumann, Joern Ploennigs, and Bernard Gorman. 2014. Towards automating the deployment of energy saving approaches in buildings. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings.* ACM, 164–167.

[24] Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing.* Association for Computational Linguistics, 1070–1079.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[26] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering* 26, 8 (2014), 1819–1837.