Fall 2023

Sensor Relationship Inference in Single Resident Smart Homes Using Time Series

Samuel Nack
Missouri State University, Nack59@live.missouristate.edu

SENSOR RELATIONSHIP INFERENCE IN SINGLE RESIDENT SMART HOMES USING TIME SERIES

A Master's Thesis

Presented to

The Graduate College of

Missouri State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science, Computer Science

By

Samuel Nack

December 2023

SENSOR RELATIONSHIP INFERENCE IN SINGLE RESIDENT SMART HOMES USING TIME SERIES

Computer Science

Missouri State University, December 2023

Master of Science

Samuel Nack

ABSTRACT

Determining sensor relationships in smart environments is complex due to the variety and volume of time series information they provide. Moreover, identifying sensor relationships to connect them with actuators is difficult for smart home users who may not have technical experience. Yet, gathering information on sensor relationships is a crucial intermediate step towards more advanced smart home applications such as advanced policy generation or automatic sensor configuration. Therefore, in this thesis, I propose a novel unsupervised learning approach, named *SeReIn*, to automatically group sensors by their inherent relationships solely using time series data for single resident smart homes. *SeReIn* extracts three features from smart home time series data - Frequent Next Event (FNE), Time Delta (TD), and Frequency (FQ). It then applies Spectral Clustering, K-Means clustering, and DBSCAN to group the related sensors. The application of unsupervised learning enables this approach to operate anywhere in the smart home domain regardless of the sensor types and deployment scenarios. *SeReIn* functions on both large deployments consisting of around 70 sensors and small deployments of only 10 sensors. Evaluation of *SeReIn* on real-world smart home datasets has shown that it can recognize inherent spatial relationships. Using three different unsupervised clustering evaluation metrics: Calinski-Harabasz Score, Silhouette Score, and Davies-Bouldin Score, I ensure that *SeReIn* successfully builds clusters based on sensor relationships.

KEYWORDS: IoT, relationship inference, sensor, smart home, time series, unsupervised learning

SENSOR RELATIONSHIP INFERENCE IN SINGLE RESIDENT

SMART HOMES USING TIME SERIES

By

Samuel Nack

A Master's Thesis
Submitted to the Graduate College
Of Missouri State University
In Partial Fulfillment of the Requirements
For the Degree of Master of Science, Computer Science

December 2023

Approved:

Razib Iqbal, Ph.D., Thesis Committee Chair

Siming Liu, Ph.D., Committee Member

Mukulika Ghosh, Ph.D., Committee Member

Julie Masterson, Ph.D., Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to firstly thank Dr. Razib Iqbal for the opportunities he gave me to do research. Without him, I would never have considered doing graduate studies, let alone publishing a thesis. In addition, his constant mentorship and amazing patience have been instrumental in ensuring the success of my graduate studies. All told, he has changed my life entirely for the better. I would also like to thank Dr. Siming Liu, Dr. Ajay Katangur, and the entire computer science department for their incredible support throughout the course of my studies at Missouri State. They have all supported my interest in whatever topic I chose to pursue. Finally, I would like to thank all the computer science graduate students for their friendship and willingness to help in whatever way they can.

I would also like to thank my parents, Robert Nack and Michelle Nack. My father's unending passion and lifelong thirst for knowledge has constantly inspired me in my own pursuits, and he more than anyone has demonstrated for me the importance of education and learning. My mother's encouragement and unwavering confidence in my inevitable success has sustained me through the most difficult and challenging endeavor of my life. My parents have always been my first teachers, and without their example and their support I would never have made it this far.

This thesis is dedicated to my fiancé Mya Kemper.

TABLE OF CONTENTS

Introduction 1
- Motivation 1
- Research Challenges 3
- Research Questions 4
- Research Contribution 4
- Thesis Outline 6

Literature Review 7
- Sensor Location Estimation 7
- Time Series Feature Extraction 9
- Sensor Relationship Inference in Smart Buildings 10

Methodology 13
- Data Collection 13
- Feature Extraction 16
- Clustering 23

Experimental Setup 25
- Datasets 25
- Evaluation Metrics 31

Results and Discussion 38
- Calinski-Harabasz Score 39
- Silhouette Score 43
- Davies-Bouldin Score 48
- Custom Dataset 52

Conclusion 55

References 58

Appendices 61
- Appendix A: Calinski-Harabasz Boxplots 61
- Appendix B: Silhouette Score Boxplots 64
- Appendix C: Davies-Bouldin Boxplots 67

LIST OF TABLES

Table 1 Summary Statistics for the CASAS dataset 29

Table 2. Description of the Custom Dataset 31

Table 3. Summary of Parameters for Feature Extraction Techniques 39

Table 4. Calinski-Harabasz Score Mean for Frequent Next Event. 40

Table 5. Calinski-Harabasz Score Mean for Time Delta. 41

Table 6. Calinski-Harabasz Score mean for Frequency. 42

Table 7. Silhouette Score mean for Frequent Next Event. 44

Table 8. Silhouette Score mean for Time Delta. 45

Table 9. Silhouette Score mean for Frequency. 47

Table 10 Worst Sensor by Category According to Silhouette Score. 48

Table 11. Davies-Bouldin Score mean for Frequent Next Event. 49

Table 12. Davies-Bouldin score mean for Time Delta. 50

Table 13. Davies-Bouldin score mean for Frequency. 51

Table 14. Custom Dataset Clustering Results 53

Table 15. Scores Using Frequency on Custom Dataset 54

LIST OF FIGURES

Figure 1: Overview of the proposed approach. 13

Figure 2: A Smart Home Recording Data from Sensors as a Time Series. 15

Figure 3: Pictorial representation of the FNE algorithm. 19

Figure 4: Pictorial representation of the TD algorithm. 22

Figure 5: Some example maps from the CASAS dataset showing testbed layouts. 27

Figure 6: An example text file from the CASAS dataset showing several sensor events. 28

Figure 7: Layout of the custom dataset collection apartment. 31

Figure 8: Some examples of clustering algorithm results on the CASAS dataset. 33

Figure 9: FNE and TD Clustering Results for Custom Dataset. 53

Figure 10: FQ Clustering Results for Custom Dataset using weekday, hour, and minute bins. 54

LIST OF EQUATIONS

Equation 1: Within Group Sum of Squares. 34

Equation 2: Between Group Sum of Squares. 34

Equation 3: Calinski-Harabasz Score. 34

Equation 4: Average Within Cluster Distance 35

Equation 5: Average Between Cluster Distance. 35

Equation 6: Sample Silhouette Score. 35

Equation 7: Population Silhouette Score. 35

Equation 8: Cluster Similarity. 36

Equation 9: Davies-Bouldin Score. 36

INTRODUCTION

Motivation

With the advancements of Internet of Things (IoT) technologies, the variety and volume of smart devices have rapidly increased. According to the International Data Corporation, worldwide Internet of Things spending is expected to surpass $1 trillion in 2026 [1]. This spending leads to more and more devices, with IoT Analytics projecting 16.7 billion devices by the end of 2023 [2]. Generally, these devices can be broken down into two categories. Firstly, there are sensors, which can measure various things about their environment such as light, noise, or temperature. Then there are actuators, which physically interact with their environment to cause changes, whether by directing electrical power to a device in the case of smart lightbulbs or moving a motor like the pin in a smart lock. Smart environments consist of both sensors and actuators deployed to a physical location to improve inhabitants' experience of that environment [3]. These smart environments can be smart rooms, smart homes, smart buildings, or even smart cities. Depending on their scope and inhabitants, each smart environment can have different specific goals related to improving user experience. One smart environment might focus on safety, while another could attempt to save its inhabitants' money by reducing energy usage. In this thesis, I consider smart homes, which aim to make the lives of their residents easier by automating common household tasks.

Today, smart homes like Amazon's Alexa Smart Home can connect multiple devices together to automate tasks like temperature control, turning lights on or off, and locking doors [4]. However, as noted above, the number of smart devices is only growing, as well as the

capabilities of these devices. Samsung's Smart Things page lists several new technologies, like AI powered clothing washing that detects soil levels that automatically determine the necessary amount of detergent or robot vacuums that avoid pets based on their location [5]. However, in order for this explosive growth in the domain to result in actual improved experience for the end user, the infrastructure surrounding smart homes must be prepared for this deluge of new devices. Currently, Samsung's Smart Things limits the number of devices per account to 200, and while this seems like a large limit for now, in light of the projection by IoT Analytics of nearly 17 billion devices in coming years, this limit is not enough. Not only does the limit on the number of devices need to be expanded, but also new systems need to accommodate the variety of new types of sensors. To effectively utilize these myriad devices, it is necessary to first obtain contextual information [6]. This can include details about the purpose of a smart device, the data it reports, its physical location in a home, or its relationships with other devices.

In this thesis, I focus on sensor relationships, a form of sensor context information that describes how an individual sensing device exists within a smart home made up of many sensors. Each sensor has logical connections to many other sensors throughout a smart home, and that relationship information can help determine the function of the device as well as its location. To further narrow the focus of this work, I explicitly focus on identifying spatial relationships, since sensors that exist in the same area can affect each other. Currently, frameworks like Amazon's Alexa can facilitate connecting new devices to the smart home [4]. However, this does not identify sensor relationships, as it only connects new devices into a central hub and does not identify the other devices to which the new device is related. To the

best of my knowledge, the commonly prevailing method to obtain sensor relationship information is by manually labeling each device with a list of other devices it is related to, a process which will be discussed in greater detail in the related works section. Unfortunately, any human process is subject to failure due to error. In addition, manual labeling is unscalable thanks to the quickly growing number of devices worldwide. As the number of devices grows the number of possible relationships, and therefore the number of manual labels required, grows quadratically [7]. Further, any change to a smart home can require a change in the labeling of all the sensors, which would add to the already undesirable amount of work required. Therefore, it is evident that there is a need for an automatic system to gather sensor relationship information.

Research Challenges

Any approach that seeks to automate the sensor relationship inference process generally takes two forms [6]. Firstly, the metadata driven approaches, which seek to parse human-created labels on sensing devices. This simply shifts the problem of requiring human labeling from the relationship inference domain to the sensor setup domain and has the same problems described in the motivation section above. Secondly, there are data-driven approaches, which analyze time series data. According to Langkvist et. al., a time series is a representation of a continuous process, such as the reports of a sensing device, as a list of discrete points. Time series analysis can have high complexity due to the noisy and high-dimensional nature of time series data [8]. Again, due to the rapid growth in the number of devices, a low complexity approach is extremely important.

As a further constraint in the smart home domain, since these devices may be employed by people without an extensive technical background, any automatic inference approach cannot be too complex or require too much computational power. A complex system would potentially confuse smart home users who have little technical experience, and a computationally heavy approach would require powerful devices which would not be cost effective for smart homes. Therefore, a low complexity, automatic inference approach that does not rely on extensive expertise or manual labeling by the end user is required.

Research Questions

In this thesis research, I have focused on three research questions:

1. Can I determine smart home sensor relationships without a labeled dataset?
2. Can I determine smart home sensor relationships based entirely on the time series data they report?
3. Can I extract features from time series data that summarize information about sensor relationships?

My hypothesis is that an unsupervised learning approach can be designed to perform feature extraction on smart home time series data in order to learn the relationships between sensors, and group them based on these relationships into spatially proximate sensor clusters.

Research Contribution

To prove my hypothesis, I propose *SeReIn*, a novel *Se*nsor *Re*lationship *In*ference approach, to automatically determine the spatial relationships based on time series data.

SeReIn extracts information from sensor data to reduce the dimensionality and noise inherent in time series data. This comes in the form of three feature extraction techniques specifically designed for the sensor relationship inference in smart home domain, namely: frequent next event (FNE), time delta (TD), and frequency (FQ). Each feature extraction technique is specifically designed for the smart home sensor relationship inference domain to capture information relevant to sensor relationships while ignoring unnecessary information.

FNE and TD determine what sensors commonly respond to the same phenomenon by determining which sensor events happen in close succession, while FQ finds which sensors have similar data patterns. These features reduce the dimensionality of the data by creating adjacency matrices based on how different sensors all commonly respond to the same human activity. *SeReIn* uses unsupervised learning, meaning that it seeks inherent structure in the data without learning from some ground truth labels [9]. This unsupervised learning comes in the form of three clustering algorithms, namely spectral clustering, K-Means clustering, and Density Based Spectral Clustering of Applications with Noise (DBSCAN). Once clustered, I use three evaluation metrics created for unsupervised learning, namely: Calinski-Harabasz Score, Silhouette Score, and Davies-Bouldin Score. These three evaluation metrics ensure that the clusters built by *SeReIn* actually determine sensor relationships and also that the clusters themselves are well-formed.

This thesis work has resulted in the following peer reviewed publication: S. Nack, R. Iqbal and S. Liu, "SeReIn: Smart Home Sensor Relationship Inference," *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM),* pp. 1-6, 2023.

Thesis Outline

The rest of this thesis is organized as follows: the Literature Review section familiarizes the reader with similar fields and closely related works. The Methodology section describes *SeReIn*, including detailed description of the feature extraction techniques. The Experimental Setup section details the datasets and metrics I used to determine the effectiveness of *SeReIn* in real smart home scenarios. The Results and Discussion section shows the performance of *SeReIn* on the experiments described in Experimental Setup, and the Discussion explains the relevance of these results. The Conclusion section summarizes the entirety of the paper and suggests directions for future work.

LITERATURE REVIEW

In this section, I briefly cover existing works in three research areas related to *SeReIn*, namely sensor location estimation, feature extraction on time series data, and application of sensor relationship inference in fields outside of smart homes.

Sensor Location Estimation

Firstly, sensor location estimation is the process of identifying where in space a sensor exists based on its data transmission without an explicit measurement. With large deployments, manually measuring and specifying the location of each sensor is not scalable, and prone to human error. Sensor location estimation enables smart environments to determine the context of a given sensor and more effectively utilize its capabilities. There exist several techniques to approximate the location of a wireless sensor. Global Positioning Systems (GPS), can be used to determine sensor locations, however GPS is not built for indoor use cases like smart homes. Therefore some researches have turned to other alternatives.

In [10], the authors used signal strength of radio waves to estimate sensor location within a few meters. However, this work requires extra base stations to triangulate sensor location, which is undesirable in the smart home domain due to the extra cost and complexity associated with additional device requirements. In [11], the authors perform location estimation in a ZigBee network based on fingerprinting techniques, which essentially seeks to uniquely identify devices. This is more desirable as ZigBee protocol is often utilized in smart homes, however, the use of base stations is still required. The signal triangulation and ZigBee

signal strength measurement approaches can effectively identify the approximate location of a mobile sensor.

In [12], the authors attempt to estimate the distance between a wireless access point and a smartphone device in order to improve wireless local area networks. Their approach uses a Recurrent Neural Network (RNN) based architecture on data gathered from smartphone inertial measurement units (IMU). The authors created a custom dataset to train their model by developing a smartphone application to read mobile phone sensors, including an accelerometer, gyroscope, magnetometer, and orientation sensor. They then asked participants to walk along a predefined path through a building multiple times. When testing their model, they were able to accurately estimate distance of the user from the access point in real time with an average error of 1.26% over distances up to 400m.

This spatial location or distance estimation can be used to group sensors, however these approaches do not take into account physical barriers that may translate into logical divides in sensor groups. For example, two sensors can be 10ft away from each other but on different floors, or different sides of a thick wall, or on opposite sides of a single room. These are all extremely different kinds of distances when dealing with sensor relationships, but the approaches listed above do not deal with this problem. *SeReIn* can group sensors spatially, in such a way that takes these barriers into account. In addition, the distance estimation specifically relies on combined data from several different sensing devices in order to track a moving smartphone. This would not be applicable to the smart home domain, as not every sensor moves, nor is it feasible to attach multiple movement sensors to every smart home sensor.

Time Series Feature Extraction

Secondly I deal with time series feature extraction. Time series data has high dimensionality and is often subject to noise [8]. This means that analysis of time series data can be a lengthy process as the number of dimensions creates more computations. In addition, there is noise in the data, which means that not all time series events are representing important information. Some are representing false, redundant, or unnecessary data. Feature extraction in time series reduces noise and speeds up analysis by reducing the number of features [13].

Many works spanning decades exist that seek to extract meaningful features from time series across various domains. In the fields of speech recognition, musical genre recognition, and video processing, features such as Mel-frequency Cepstral Coefficients, Chroma, Scale Invariants Feature Transform, and Histograms of Oriented Gradients have been identified as having particular relevance to their respective fields [14], [15]. These features based on time series data have proven useful for their respective domains, but are not always as useful outside the specific fields for which they are designed. Rather than using explicitly defined features, Yazdi et al. in [16] propose an unsupervised learning to find inherent structures present in the data. They use a time series specific sparse coding approach to create a small dictionary of a few functions in such a way that all data in a given input can be represented as a linear combination of those functions. Their approach reduces all time series in a dataset to the same size as the dictionary, however, it has time complexity of $O(T^2N)$ where T is the length of the time series and N is the number of sensors. Therefore, analyzing time series data in the smart home domain requires that I design new features, since to the best of my knowledge, no

widely adopted domain-specific features for sensor relationship inference exist outside of our proposed approach.

Sensor Relationship Inference in Smart Buildings

Finally, I explored sensor relationship inference in the smart building domain. Smart buildings are another kind of smart environment, but rather than dealing with residential homes, smart buildings tend to deal with larger structures such as offices. Smart homes focus on conforming as closely as possible to the comfort and convenience of a small group of people. In contrast, smart buildings must accommodate hundreds of people across several floors by combining information from thousands of smart devices. They focus more on general needs and efficiency rather than trying to align closely to individual desires.

Some approaches seek to use sensor metadata to to identify sensor relationships. Metadata is data about data, or information that is specifically created and used to describe certain aspects of things like sensors [17]. In [18], the authors used string matching to parse sensor metadata for energy management systems. They created a dictionary of possible acronyms or strings they might come across, then compared the metadata from IBM's research living laboratory bulding in Dublin. Each metadata string would find the items in the dictionary to which it was the most simila. However, this system would only rank the most likely labels, and in only 16% of the testing set were the dictionary entries wth the highest similarities actually the correct match. In addition, this approach relied on certain metadata standards, which are sets of rules governing how metadata strings can be formed. Specifically, this example relied on predetermined separator values between fields in the metadata string.

Depending on who deploys a given sensor, they may personally adhere to a different metadata standard, or perhaps be inconsistent in their adherence to any standard. To counteract this, the authors in [19] develop a standard agnostic system to parse smart building sensor metadata. Their goal is to convert different kinds of standards to a common representation. Their system doe this by first grouping sensors based on metadata with similar structures. It then present a metadata tag to an expert who describes what each part of the tag means. Using the expert parsing, the system finds new sensors to present to the expert so that it can learn rules in the form of "if then else" statements that can transform the metadata tags from the unknown standard into the common standard. The authors test their approach on three large smart buildings with 1586, 2522, and 1865 sensors respectively, and they are able to classify 70% of sensors with just 24, 15, and 43 expert-parsed examples. However they encountered issues with obscure tags which referenced devices that were unique or not often used.

Metadata approaches cannot fully enable automatic sensor relationship inference. Firstly, they require human input to originally create the metadata. In addition, they will only identify relationships that a human saw fit to identify in the first place. Sensor groupings that are not immediately obvious to the human labeler or ones that are simply missed due to human error would then be lost. Therefore other research has focused on identify sensor relationships based instead on the data reported by the smart devices.

For example, in both [20]and [21], the authors perform manual changes to the physical system's settings to collect relevant data, which requires domain knowledge about the specifics of the systems in a given smart building. In [6], Li. et. al. use a convolutional neural network to learn a comparison metric for time series data that can distinguish between related and

unrelated sensors. This alleviates some of the required domain knowledge and time requirements of previous approaches, since the network can determine the comparison metric regardless of the type of relationship between sensors. However, a labeled dataset requirement forces their approach to only learn relationships that are predefined or labeled, meaning the model may not pick up on any implicit groupings in unique deployments. In comparison, *SeReIn's* sensor relationship inference approach has no metadata or labeled dataset requirements.

METHODOLOGY

There are three primary steps for *SeReIn* to identify sensor relationships in a smart home as shown in Figure 1. Sensor data is first collected and organized into a single time series. Next, my novel features frequent next event (FNE), time delta (TD), and frequency (FQ), are extracted in order to simplify the time series in such a way that preserves sensor relationship information. Then, I apply clustering method on the extracted features to cluster data according to their similarities determined by the feature extraction. In the following sections, I examine each of the three components specified in Figure 1 in greater detail, and explain their sub-components.

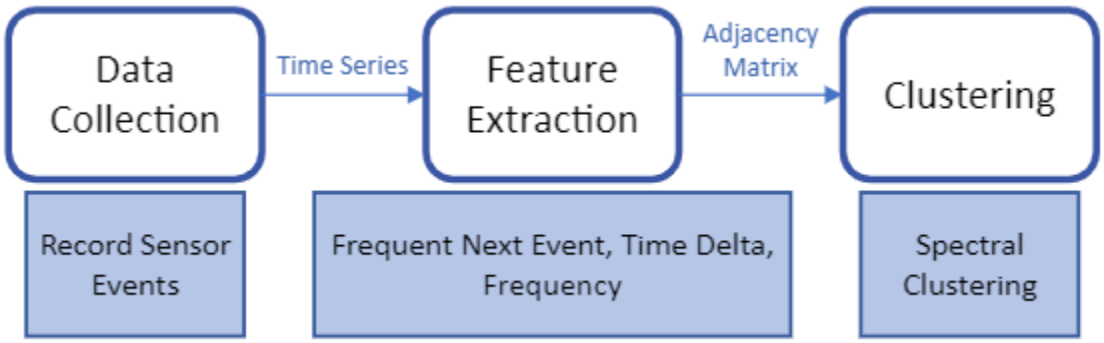


Figure 1: Overview of the proposed approach.

Data Collection

In order to analyze a set of smart home sensors, it is important to first collect and organize the data that they report. In this section, I define sensor events and explain the role they play in *SeReIn's* analysis. For the purpose of this work, I define a sensor event as a state

change in any sensor in a given smart environment. I consider any change in a sensor value to be an event, regardless of the direction of that change, or what the sensor is measuring. I do this to alleviate the amount of domain knowledge and manual labeling required to fine-tune the system for a given smart environment. For example, if I treated the motion off event as different from the motion on event, *SeReIn* would then be forced to require all its users to label which sensors are motion sensors. Or if I wanted to have different sensors be treated differently, I would need some sort of understanding about the function of each sensor, not just in general, but in the specific smart home in which it is deployed. This knowledge, while valuable, is difficult to obtain, and one of the major goals of *SeReIn* is to alleviate this domain knowledge requirement. Therefore, I always consider sensors events as any change in a sensor value, regardless of the direction or magnitude of that change.

Sensor events consist of three vital components: *event_time* which is the exact time at which the event occurred, *sensor_ID* which is the unique name that identifies every sensor, and *data_value* which is the new value of the sensor that triggered the event. I record these three elements of every event, and over time I amass a series of time-labeled events. Thus, a smart home can be thought of as a time series of sensor events, collected over time as a user interacts with their home. This is show in Figure 2. As sensors detect changes in their environment, they will eventually collect a sizeable amount of data. In this work, I mainly deal with historical data, meaning that I consider only a subset of smart home events that have happened in the past. *SeReIn* could be adapted to include live data, where it continues to update its sensor groupings in real-time as new events are reported. I decide instead to focus

on historical data to determine the efficacy of this approach and leave live-prediction as future work.

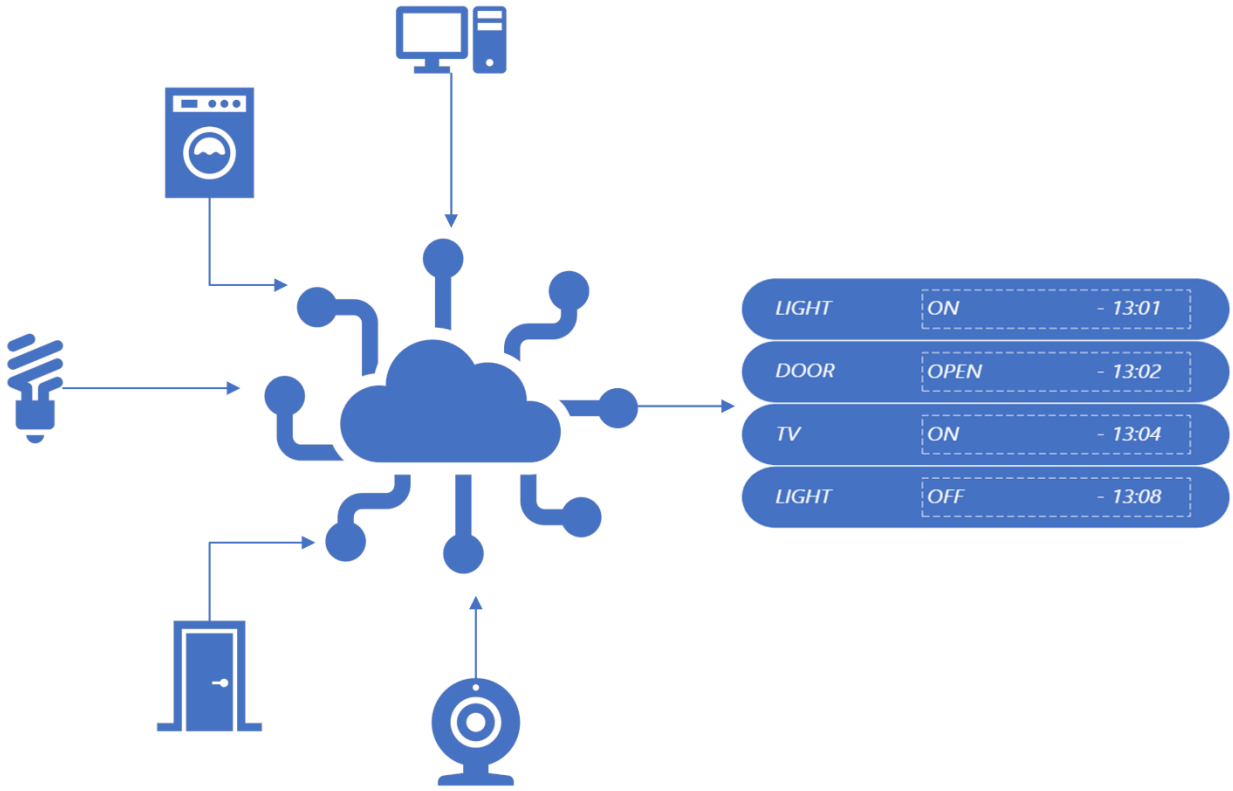


Figure 2: A Smart Home Recording Data from Sensors as a Time Series.

Depending on the exact size of the smart home and the time interval of historical data being considered, the sensor event time-series can easily grow to millions of records. Interspersed randomly throughout these large time series are noise events, which are any events caused by randomness rather than an actual change in a sensor state. They can be caused by random electrical episodes, sensor glitches, or duplicate data reporting due to network issues. These noise events as well as the sheer number of regular events quickly make

any time series analysis difficult. Therefore, *SeReIn* extracts key features from the time series to counteract the high dimensionality and noise while still retaining information about sensor relationships.

In this section, I defined how any smart home can be organized into a single time series. In the Experimental Setup section, I describe two real-world smart home datasets that I organized into time series to establish the effectiveness of my approach. More details on the actual data is found in Experimental Setup, subsection Datasets.

Feature Extraction

SeReIn uses three novel features, FNE, TD, and FQ, that encode different aspects of sensor relationships solely from the time series data. More specifically, FNE and TD both take into account the context of what sensor events are commonly occurring in quick succession. FQ considers frequent usage patterns to identify when sensors are being used together. The extraction of these three features results in three separate adjacency matrices, which efficiently encode the similarities between sensors in a square matrix. This scheme eliminates the requirement to store millions of records from historical time series data, and it speeds up our clustering by reducing the dimensionality of its input.

In the following sections, I give the formulation of each of these feature extraction methods. This is done through algorithms, descriptions, and step-by-step demonstrations of the feature extraction at work. In addition, I will explain the insight behind each of these methods through hypothetical examples involving smart home sensor relationships.

Frequent Next Event (FNE) Feature. FNE is calculated by identifying which sensor events frequently follow each other. The algorithm used to calculate this feature is outlined in Algorithm 1 and represented graphically in Figure 3. Firstly, I create a square matrix of size *n x n*, where *n* is the total number of unique sensors in a given smart home. Then, the entire time series is iterated over one event at a time and, at each iteration, the next event is considered. I update a single entry of the square matrix where the row corresponds to the current sensor event and the column corresponds to the next event being considered. The magnitude of this change is a function of the time difference between the two sensor events.

I explored three different increment methods: constant, time-based, and time-based with rounding. The constant method will always return a one, therefore regardless of the time difference between two sensor events the increment value is always one. The time-based method does take into account the time difference, and follows the formula $1/t$, where the increment value has an inverse relationship to the time difference between the sensor events *t.* This means that values close to zero will result in a large increment value, while larger values of *t* will result in smaller increments to the adjacency matrix. The time-based with rounding method is the same, but simply rounds the time difference *t* up to the nearest whole value. This is to prevent values of $t < 1$ from rapidly causing large increment values, however, I test both time-based and time-based with rounding to identify whether this intuition is valid. I evaluate the efficacy of these three methods in the Results and Discussion Section.

Algorithm 1 FNE Calculation

Require: D = [A list of sensor events] & inc = a function of time
$currentEv = D[0]$
$adjacencyMatrix$ = zeros($size = (n \times n)$)
for nextEv in D **do**
  if $nextEv.ID \neq currentEv.ID$ **then**
    t = ceil($nextEv.Time - currentEv.Time$)
    $adjacencyMatrix$[currentEv.ID][nextEv.ID] += inc(t)
  end if $currentEv = nextEv$
end for
return $adjacencyMatrix$

This feature captures relationships where events in one sensor commonly follows events in another. For example, in Figure 3, I consider how FNE would evaluate a series of smart home events recorded by a motion sensor and a temperature sensor. Suppose this data describes an individual walking into the room and turning on a heater, increasing the temperature. A relationship exists between the motion and temperature sensors, and this relationship would be captured by the FNE feature. On the left side of the figure is the smart home dataset, while on the right is the adjacency matrix calculated by FNE. Red represents the previous event, and blue represents the next event. FNE iterates over the entire smart home time series two events at a time. In Figure 2, FNE first consider a motion sensor event followed by a temperature sensor event. From the adjacency matrix, it selects the row corresponding to the motion sensor, since that is the previous event. Then FNE finds the intersection between that row and the next event, which is the temperature sensor. Then it increments the adjacency matrix at that intersection, in this example using the time-based increment method. Since the

two events are 1 second apart, this increments the adjacency matrix by 1. Next, FNE considers the next two values, which correspond to the temperature sensor and the motion sensor, which have events two seconds apart. Therefore, FNE increments the adjacency matrix at the row corresponding to the temperature sensor and the column corresponding to the motion sensor by *1/t,* or 0.5. Finally, FNE considers the last two events in the sample dataset, which both come from the motion sensor. Events originating from the same sensor do not result in an increase in the adjacency matrix, therefore nothing happens on this iteration.

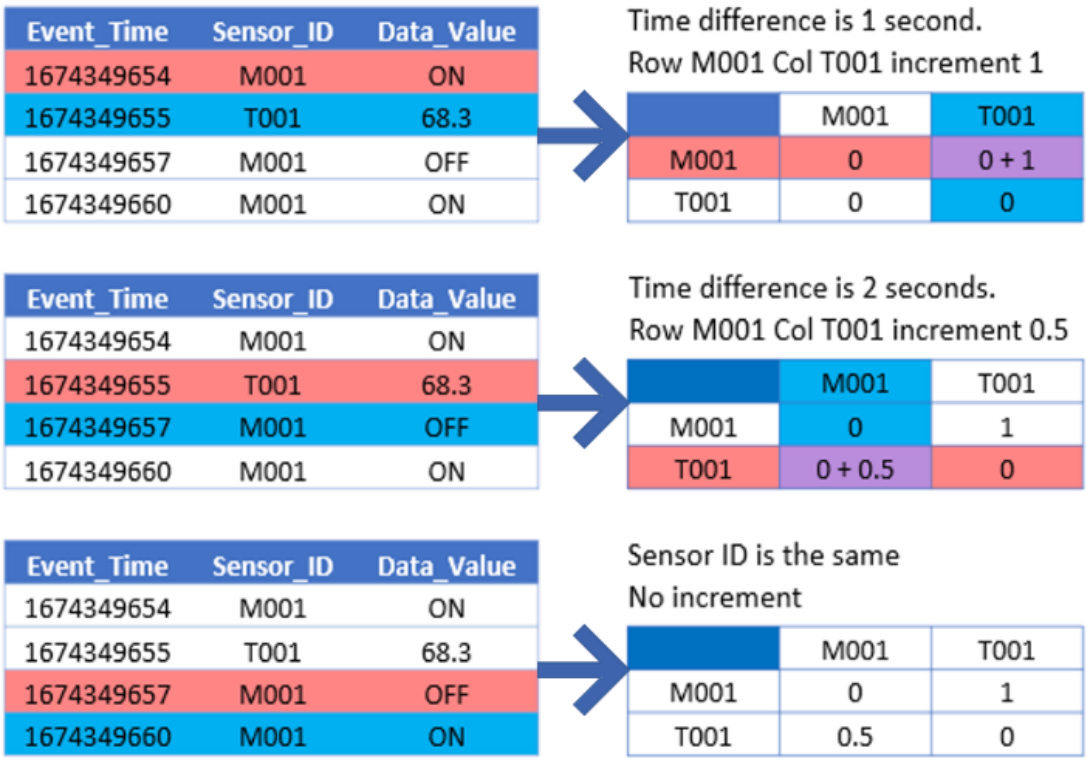


Figure 3: Pictorial representation of the FNE algorithm.

Time Delta (TD) Feature. TD groups sensor events using the amount of time that has elapsed between sensor events. I outline the algorithm used to calculate this feature in Algorithm 2. Similar to FNE, the final result of this method is an adjacency matrix of size *n x n*,

where *n* is the total number of unique sensors. To calculate this adjacency matrix, the algorithm breaks the sensor events into groups. At each iteration, it determines if the next sensor event is part of the current group by calculating the amount of time between the current sensor event and the next sensor event. If this time difference is under a certain threshold *th,* then I consider the sensor as part of the group, and add the sensor's ID to the group if it is not already present. I then increment the adjacency matrix by one in the column of the given event and the row of every unique sensor in the group. I experiment with the effect of this threshold value in later sections, using values of 1 second, 2 seconds, 3 seconds, and 5 seconds. Larger threshold values would enable a wider variety of sensors to form groups using this method. However, larger threshold values would also be more subject to interference by noise.

For example, in Figure 4, I again consider a motion sensor and temperature sensor. On the left is the smart home dataset, while on the right is the adjacency matrix calculated by TD. Red represents the previous event, and blue represents the next event. For this example, the threshold value *t* is set to 3 seconds, so any events that are 3 or more seconds apart will be split into new groups. Suppose this data describes an individual who walks into a room, turns on the heater, then leaves the room to go to the bathroom. This is shown in the time series data by a motion on, a heat increase, and a motion off, then a short time later another motion on event. The TD feature recognizes the events taking place within a threshold as part of one grouping, whereas the final event in our example of leaving the room falls outside this threshold and is therefore part of a new group.

Algorithm 2 TD Calculation

```
Require: D = [A list of sensor events] & th = an integer
currentEvTime = D[0].Time
adjacencyMatrix = zeros(size = (n × n))
currentGroup = [ ]
for nextEv in D do
    if nextEv.Time − currentEv.Time < th then
      if nextEv.ID not in currentGroup then
        currentGroup.append(nextEv.ID)
      end if
      for item in currentGroup do
          adjacencyMatrix[item][nextEv.ID] += 1
      end for
    else
      currentGroup = [ nextEv.ID]
    end if
    currentEvTime = nextEv.Time
 end for
```

Like FNE, TD iterates over the entirety of the smart home dataset two sensors at a time, with the exception being the first iteration. On the first iteration, TD simply analyzes the motion sensor event as the next event and assumes there is no previous event. Since the group created by TD has no members initially, TD adds the motion event being considered to the current group. Then, TD at each iteration considers the rows corresponding to every sensor in the current group, which on this iteration means that only the row corresponding to the motion sensor is considered. Next, TD intersects these rows with the column corresponding to the next event, which is again the motion sensor in this case. TD increments this intersection by 1 before moving on to the next pair of events. On the next iteration, TD considers the motion event as

the current event and the temperature event as the next event. Since the time difference between these two events is within the threshold value of 3 seconds, TD adds the temperature sensor to the current group. The rows corresponding to the current group, which in this case is both the temperature and motion sensors, are intersected with the column for the next event, which here is the temperature sensor. This intersection is the incremented.

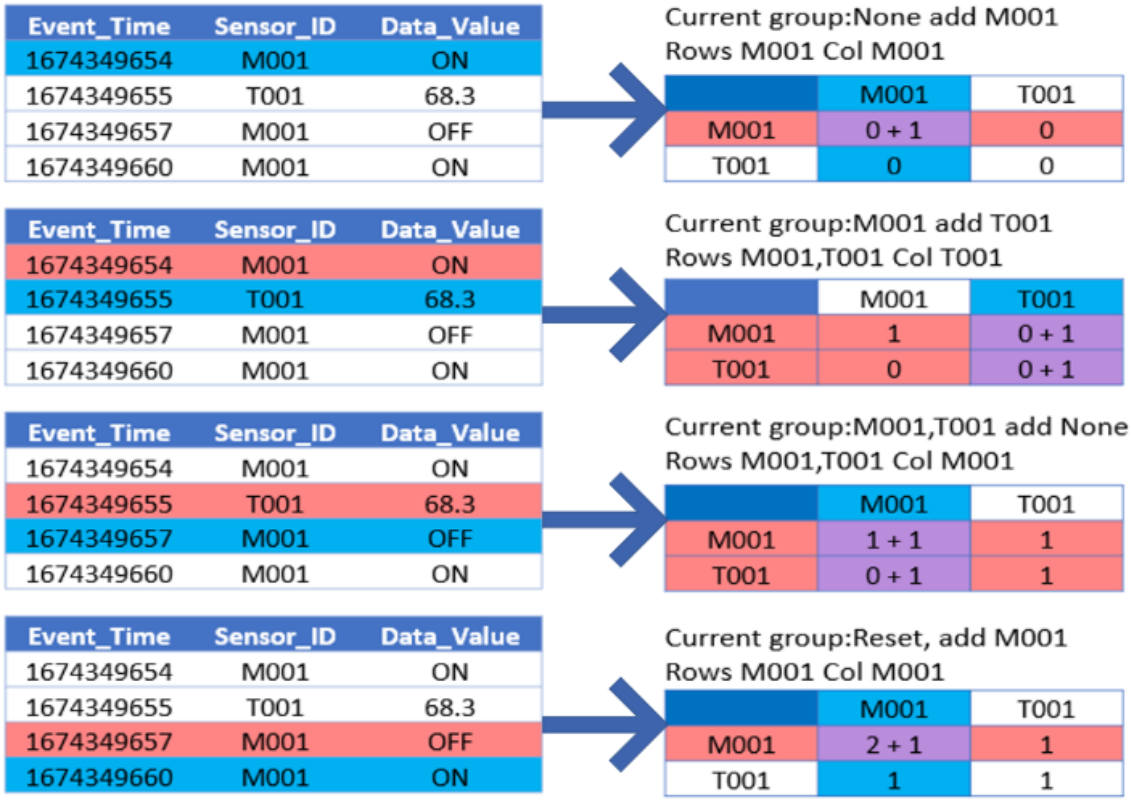


Figure 4: Pictorial representation of the TD algorithm.

For the next iteration, TD considers the temperature sensor as the previous event and the motion sensor as the next event, and since these two events are within the threshold value it continues with the current group. Additionally, since the motion sensor is already in the current group, there is no need to add anything to the current group. Therefore, TD intersects

the motion and temperature sensor rows with the motion sensor column and makes the necessary increments before moving on to the next iteration. At this point, the two motion events being considered are three seconds apart, which is outside the threshold. TD then discards the current group and makes a new group with next event, here being the motion sensor. Finally, TD increments the entry at the intersection of the motion sensor row from the current group with the motion sensor column corresponding to the next event.

Frequency (FQ) Feature. FQ analyzes at what times a given sensor tends to commonly detect events. This is accomplished by counting, for each sensor, the total number of times an event occurs within a given interval of time. This can be any suitable unit of time, andI experiment with different time intervals, including minute, hour, and weekday in later sections. The result of this analysis is an $n \times \tau$ matrix, where n is the number of sensors and τ is the number of time intervals or binning values by which I separate events. I then convert this matrix into an $n \times n$ matrix by measuring cosine similarity between all rows.

Other research [6] in sensor relationship inference has found success in utilizing the Fourier Transform in order to analyze sensor time series in the frequency domain. *SeReIn* uses a simplified binning approach rather than Fourier Transforms, otherwise, every new sensor event would require a new run of the Fourier Transform, whereas with our approach I can simply increment the appropriate bin.

Clustering

In the clustering phase outlined in Figure 1, the extracted features are passed to a clustering algorithm in order to organize the sensors into groups based on their relationships. I

test three clustering algorithms, namely: spectral Clustering [22], K-Means clustering [23], and Density Based Spectral Clustering of Applications with Noise (DBSCAN) clustering [24]. The outputs of all three clustering algorithms are a list of n integer labels, one for each sensor. For spectral clustering and k-means clustering, these labels range from 0 to k, which is an argument we supply to the clustering algorithms that determines how many clusters to create. For the DBSCAN clustering algorithm, the labels can be -1 or any integer. DBSCAN creates clusters based on the $min\_samples$ and $epsilon$ or ϵ parameters. For our experiments, we hold $min\_samples$ constant at 2 and vary ϵ.

These labels determine which group each sensor belongs to, with the goal being that related sensors are placed into the same groups. I specifically chose the spectral clustering algorithm because it operates with the understanding that the data it is presented comes in the form of an adjacency or affinity matrix, which is the output of the feature extraction component. The k-mean algorithm is a commonly used clustering algorithm and was included to provide more variety to the approach, as well as provide a baseline against which to compare other algorithms. The DBSCAN algorithm was included because it does not require the number of clusters as a parameter, and I wanted to see how many clusters it would arrive at without my input. In addition, the DBSCAN algorithm uniquely uses the amount of closely packed points to build clusters, rather than centroids like k-means and spectral clustering.

EXPERIMENTAL SETUP

In this section, I introduce the two datasets I use to evaluate *SeReIn*. This involves identifying what exactly the dataset contains, as well as justifying why these datasets in particular are interesting and applicable to the research problem. Next, I explain how I plan to evaluate the clusters created by *SeReIn* both subjectively and objectively. For the objective metrics, I explain the mathematical formulation of the three evaluation metrics I used and why chose these metrics.

Datasets

I evaluate our approach on two datasets consisting of real-life deployments of sensors in smart homes. The Center For Advanced Studies In Adaptive Systems (CASAS) dataset [25] contains several different smart homes each with different sensor arrangements. In addition, I created our own dataset by deploying sensors in a two-bedroom apartment. For each smart home across both datasets, all three features defined in the Methodology section were extracted from the smart home time series data. These features were then passed to the clustering component and the results were saved for evaluation.

CASAS. The CASAS dataset was originally created by Washington State University and is widely used in the field of activity recognition. It consists of several testbeds, apartments outfitted with various sensor spread throughout the rooms. These sensors include motion sensors, light sensors, temperature sensors, and door sensors. Each testbed is equipped with anywhere between 34 and 73 sensors, which allows us to test the effectiveness of *SeReIn* on

both large and small deployments. The CASAS website lists data from many of these testbeds which can be freely downloaded. Each download contains a map of the layout of the apartment as well as the placement of the various sensors. Example maps are presented in Figure 5. It also contains a text file that lists out every sensor event reported in the apartment, along with the activity to which each sensor event belongs. Some of these files span several months and contain millions of sensor events. An example dataset text file is presented in Figure 6.

This dataset was chosen due to the wide variety of number of sensors, testbed layouts, and number of sensor events. This variety allows us to test multiple different possibilities to ensure *SeReIn* is not only accuracte in certain niche scenarios. For example, I can identify whether *SeReIn* is more effective in smaller deployments of only 30 sensors or larger ones of 70 sensors. Further, this variety is reflective of the variety of the smart home domain itself, where every single smart home is slightly different. Although it was designed for activity recognition, the CASAS layout map enables us to identify spatial relationships between sensors and ensure that *SeReIn* is accurately grouping sensors. In addition, the large amount of data ensures the statistical significance of any results. Finally, the dataset provides consistent naming schemes and organization, facilitating complex analysis. All motion sensor names start with “M”, all light sensor names start with “L”, all door sensor names start with “D”, and all temperature sensor names start with “T”.

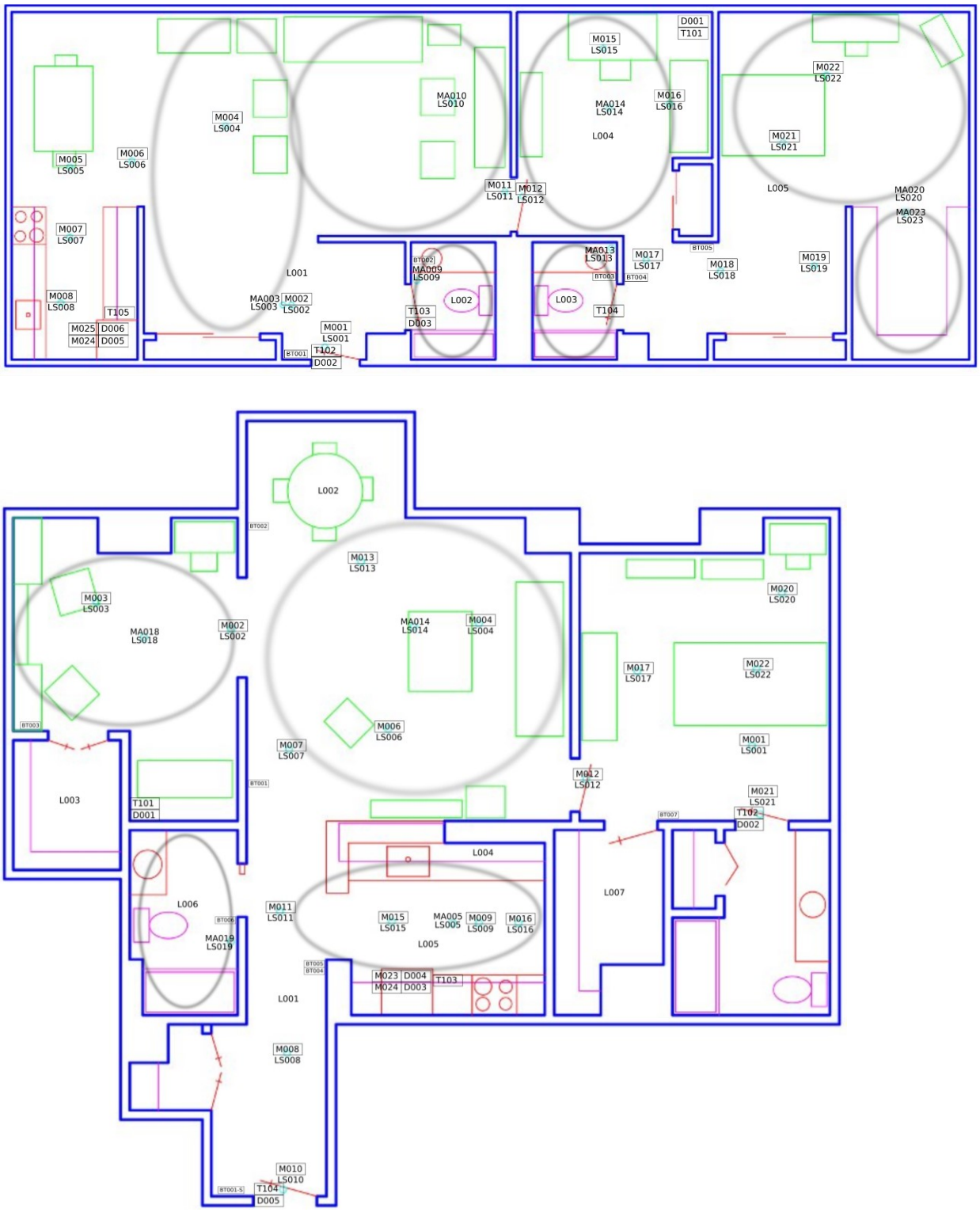


Figure 5: Some example maps from the CASAS dataset showing testbed layouts.

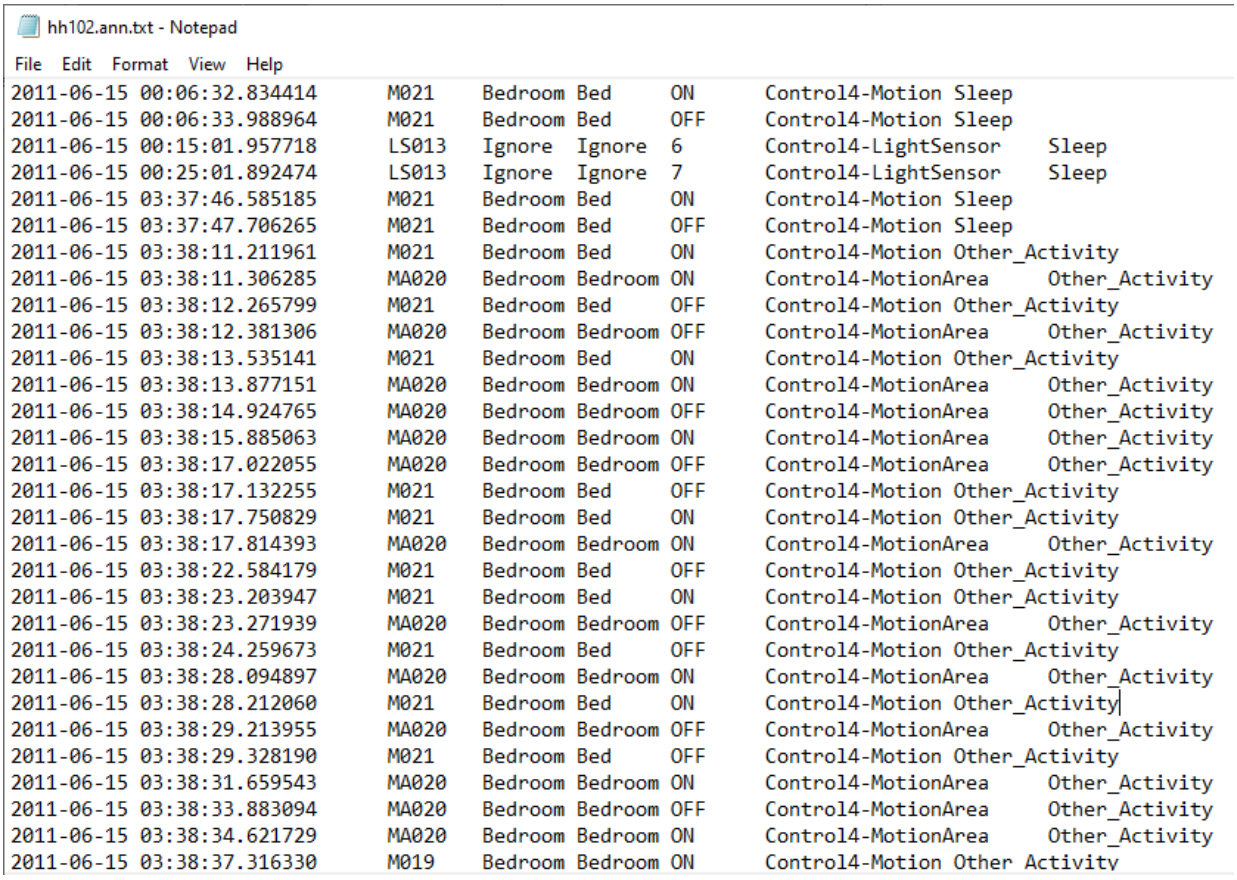


Figure 6: An example text file from the CASAS dataset showing several sensor events.

From the CASAS dataset, a total of 16 testbeds were selected for analysis. A summary of important statistics about these testbeds is given in Table 1. This wide variety in number of sensors, number of events, and even the disparity within each testbed between the sensor with the largest number of events and the sensor with the smallest number of events allows for an in-depth and complex analysis of what factors influence the effectiveness of *SeReIn*. In the experiment, I analyze the performance of *SeReIn* on all testbeds, and then, to determine why this performance was achieved, I focus on each individual testbed performance and identify patterns.

Table 1 Summary Statistics for the CASAS dataset

Testbed name	Number of Sensors	Number of Events	Largest Sensor Events	Smallest Sensor Events
hh102	65	6416283	868328	20
hh103	34	7126618	1374045	18
hh104	73	6485276	1080174	39
hh105	57	8468656	1561317	37
hh106	71	5700250	677841	287
hh108	64	10211730	1046829	41
hh109	51	14717539	1937929	62
hh110	44	156633	21409	8
hh111	68	10531104	1409386	9
hh112	43	822453	125998	10
hh113	67	3190947	558514	17
hh114	43	10029188	1978532	16
hh115	50	2228690	334835	14
hh116	47	574911	168378	2
hh117	37	1098096	137786	8
hh118	61	7979642	926295	20

Custom Dataset. The custom dataset consists of data by observing the sensor events in a 1,100 sqft two bedroom apartment. I used four Raspberry Pi devices across various areas of the apartment. A map of these devices is displayed in Figure 7 with blue dots representing individual raspberry pi devices. The specific sensors attached to each device is outlined in Table 2. Each Raspberry Pi was programmed to log any change in state of any of its sensors. These

logs were then gathered from all four devices and combined into one dataset in a format similar to that described in the CASAS dataset.

This dataset was created to ensure further variety of dataset and verify the robustness of *SeReIn* across different deployment scenarios. Having a custom deployment ensures that *SeReIn*'s performance was measured on real-world data. Additionally, having complete control over the deployment scenario allowed us to manually create clusters that I wanted *SeReIn* to identify, whereas with the CASAS dataset I had no control over sensor placement. Finally, the creation of a custom dataset can hopefully add to the field as a whole by giving future researchers more data on which to evaluate their systems.

I collected data from this deployment over the span of several months. In total, I collected 3,657,766 sensor events across the entire apartment. I also encountered several problems with this dataset. Firstly, random electrical noise and the use of other electronic devices in the vicinity of the sensors would sometimes trigger sensor events despite no chage in the actual physical value. For example, in the bathroom, turning on a fan would cause the door and light sensors to rapidly pulse between states for a few fractions of a second. These errors actually help prove the robustness of the system, since they introduce random noise and bad data. This helps show that *SeReIn* is effective in smart homes, which are never perfect and often experience errors.

Figure 7: Layout of the custom dataset collection apartment.

Table 2. Description of the Custom Dataset

Raspberry Pi	Device Location	Sensors
1	bathroom	door, light, switch
2	bedroom	light, switch
3	kitchen	light, motion
4	living	light, switch

Evaluation Metrics

To evaluate *SeReIn*, I can subjectively measure performance by plotting the obtained clusters on their original maps as shown in Figure 8. This visual inspection process is helpful

when determining the reason behind certain clusters. For example, if *SeReIn* splits certain sensors into different groups, I can then inspect the map to see that some sensors are in a closet which might not be used often while others are in and around the main bedroom. However, visually inspecting every single cluster is tedious, and it is not a rigorous and objective process. As seen in Figure 7, it is extremely difficult to effectively determine which clustering is better, or even to objectively state whether any of the three clustering results is actually successful. Therefore, in the Results and Discussion section, I will not include a visual inspection of the results, and will use mathematical measures of success.

To objectively measure the performance of the proposed approach, I need some defined metrics by which to measure the clusters. For the custom dataset, I know which sensors are connected to each raspberry pi device. Ideally, *SeReIn* would cluster sensors based on device, therefore I have ground truth labels against which I can compare the clusters. On the other hand, I do not have ground truth labels for the CASAS dataset on which to base objective evaluation. Therefore, I selected three clustering evaluation metrics that do not require ground truth labels, namely the Calinski-Harabasz (CH) score, the silhouette coefficient, and the Davies-Bouldin (DB) score.

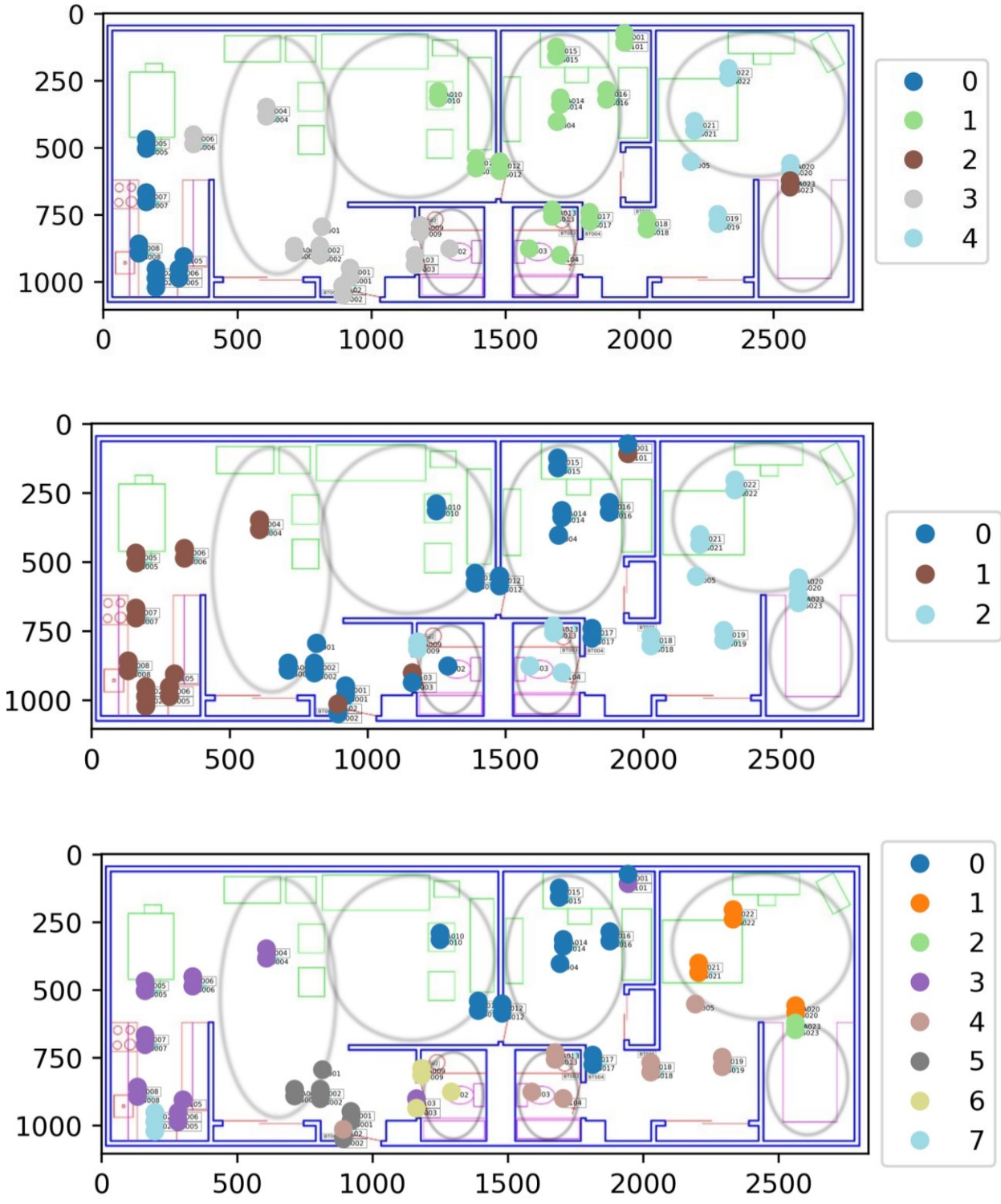


Figure 8: Some examples of clustering algorithm results on the CASAS dataset.

Calinski-Harabasz (CH) Score. The CH score [26] is computed based on the labels from the clustering algorithm and the locations given by the map and is defined as a ratio between two values. The first value is the within-group sum of squares (WGSS). This value is the sum of the distances between each point in a cluster and that cluster's center. WGSS is defined in Equation 1. The second value, defined in Equation 2, is the average distance from a given point to the absolute center or barycenter of the entire dataset, and is called the between group sum of squares (BGSS). Finally, the CH score is defined in Equation 3 as the ratio between these two values.

$$WGSS = \sum_{k=1}^{K} \sum_{i=1}^{n_k} dist(X_{ik}, C_k) \tag{1}$$

$$BGSS = \sum_{k=1}^{K} n_k \times \mathrm{dist}(C_k, C) \tag{2}$$

$$CH = \frac{BGSS}{WGSS} \times \frac{N-K}{K-1} \tag{3}$$

In Equations 1, 2, and 3 K is the number of clusters, N is the number of points, n_k is the number of points in cluster k, X_{ik} is the $i-th$ point of cluster k, C_k is the centroid of cluster k, $dist(x, y)$ is the distance by some metric between x and y and C is the barycenter of the dataset. A higher CH score indicates lower WGSS and tighter clusters as well as higher BGSS with good separation between clusters.

I chose the CH score since it takes into account both the "tightness" of each cluster via WGSS and the "spread" of the clusters via the BGSS. The "tightness" can help ensure that clusters are compact and not spread out over large regions of a smart home. The "spread"

encourages distinct clusters that are not overlapping or representing similar groups of smart devices.

Silhouette Coefficient. The silhouette coefficient [27] *s* for a single sample is defined based on two values. The first value, *a,* is the average distance between a sample and all other points in that sample's same class and is defined in Equation 4. The second value, *b*, is the average distance between a sample and all other points in the next-nearest class, and is defined in Equation 5. Using these two values, I can define *s* in Equation 6. The silhouette score for the entire set of samples *S* is then the average of all *s* in a set, and is defined in Equation 7.

$$\mathrm{A}(i) = \frac{1}{n}\sum_{j=1}^{n} dist\big(\mathrm{i}, X_j\big) \tag{4}$$

$$b(i) = \min_{i \neq j}\left(\sum_{k=1}^{K}\frac{1}{n_k}\sum_{j=1}^{n_k} dist\big(i, X_{k,j}\big)\right) \tag{5}$$

$$s(i) = \frac{b(i) - a(i)}{max(\,a(i), b(i))} \tag{6}$$

$$S = \frac{1}{N}\sum_{i=1}^{N} s(i) \tag{7}$$

In Equations 4, 5, 6, and 7, n is the number of points in one cluster, $dist(m, n)$ is the distance by some metric between m and n, i is the point I are classifying, X_j is the $j\ -th$ point of cluster X, K is the total number of clusters, n_k is the number of points in cluster k, and $X_{k,j}$ is the $j\ -th$ point of cluster X_k. The silhouette score can range from [1,-1], with 1 being the best sscore and -1 being the worst. Values near 0 are indicative of cluster overlap, while negative

values indicate misclassified samples. The silhouette score essentially measures how well a given sample fits its own cluster and compares it to how well it fits the nearest cluster.

I chose this score because it measures how well *SeReIn* separates similar sensors into their own, well-defined clusters. In addition, since this score actually measures individual sensors and then averages them into an aggregate score, I can identify which sensors had the best silhouette score and which sensors were commonly misclassified. By finding patterns in good and poor performance across testbeds, I can better pinpoint the strengths and weaknesses of *SeReIn*.

Davies-Bouldin (DB) Score. The DB score [28] is defined as the average similairty between each cluster and its most similar one. Cluster similarity is defined in Equation 8, and so the DB score is defined accordingly in 9.

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{8}$$

$$DB = \frac{1}{K}\sum_{i=1}^{K} \underset{i \neq j}{max}\, R_{ij} \tag{9}$$

In Equations 8 and 9, s_i is the average distance between each points of a cluster i and its centroid, $d_{i,j}$is the distance between two cluster centroids, and K is the total number of clusters. The minimum score for DB is 0, with low scores indicating low similarity between clusters. DB scores have no upper limit, with higher values resulting from extremely similar clusters. Ideally each grouping of sensors, and therefore each cluster, should be distinct from other clusters, which can be measured using the DB score.

I chose the Davies-Bouldin score since it has a minimum achievable score. With the CH score, it can be difficult to identify good performance since it has no limit to the scores that can

be achieved. Therefore, since the Davies Bouldin score does have a minimum valuable it can be more easily determined if a feature extraction technique is leading to suitable groupings.

RESULTS AND DISCUSSION

In this section, I separately analyze each of the three feature extraction techniques on both of the datasets presented in the Experimental Setup section. First, I perform each of the three feature extraction techniques separately on the CASAS dataset, then I perform each of the clustering techniques on the extracted feature, and then I evaluate according to our scoring metrics defined above in Section Experimental Setup. The following sections will highlight each of the three scoring metrics, each in their own subsection. These three subsections will be further subdivided by the three feature extraction techniques, with the results for each technique being displayed in a table. Every table will follow the same format, having three groups of columns, one group for each clustering algorithm. Since each feature extraction technique has multiple parameters that affect their performance, we want to compare these parameters against each other, and so each column group in the table will possess a single column for each parameter. These parameters were described in the Section Methodology, Subsection Feature Extraction, but are summarized in Table 3.

Finally, every table will have twelve rows corresponding to the parameters of the clustering algorithms. For spectral Clustering and K-Means clustering, these twelve rows will correpond to the number of clusters *k*, and will range from 2-12. For DBSCAN, the value of *min_samples* is held constant at 2, while the rows of the table correspond to the values of *epsilon*, which varies from 0.5 to 0.7 in steps of 0.02.

Following the analysis of the CASAS dataset, I analyze the custom dataset to determine if the proposed approach is effective on real world data that I collected. I will still employ the

evaluation metrics to compare the results from the custom dataset against the results from the CASAS dataset.

Table 3. Summary of Parameters for Feature Extraction Techniques

Feature Extraction Technique	Parameter Name	Parameter Value	Abbreviation
Frequent Next Event	Increment Method	constant	con
Frequent Next Event	Increment Method	time-based	tb
Frequent Next Event	Increment Method	time-based with rounding	tbr
Time Delta	Threshold	1 second	1
Time Delta	Threshold	2 seconds	2
Time Delta	Threshold	3 seconds	3
Time Delta	Threshold	5 seconds	5
Frequency	Binning Value	minute	mn
Frequency	Binning Value	hour	hr
Frequency	Binning Value	weekday	wkd

Calinski-Harabasz Score

The Calinski-Harabasz Score, as mentioned above, measures the within-cluster tightness and the between-cluster spread. In this section, I will first analyze each of the three feature extraction techniques in their own subsection, with each containing a table performance. Each table will compare CH scores when using different parameters of the feature extraction technique and different clustering methods. In Appendix A, I provide boxplots, broken down by

feature extraction method and clustering algorithm, to further detail the performance of each of these methods. At the end of this section, I include a brief discussion and result comparison.

Frequent Next Event. The overall performance of FNE according to the CH score is given in Table 4. For the spectral clustering algorithm, the time-based with rounding approach always has the highest CH score, followed by time-based, then the constant approach. For K-Means, the time-based rounding approach generally has the highest CH scores, while time-based has the lowest. For DBSCAN, the constant approach always has the highest CH score, followed by time-based with rounding, and then time-based.

Table 4. Calinski-Harabasz Score Mean for Frequent Next Event.

	Spectral			K-Means			DBSCAN			
clusters	con	tb	tbr	con	tb	tbr	con	tb	tbr	epsilon
2	12.1	16.8	25.5	6	4.8	13	2.9	1.9	2.3	0.5
3	12.4	**23**	30	**15.4**	5.7	**15.5**	3	2.1	2.5	0.52
4	12.1	18.5	30.4	11.5	**7.1**	15.1	3	2.1	2.5	0.54
5	**12.9**	18.3	**30.7**	13.6	6.1	14.3	3	2.2	2.6	0.56
6	12.5	19	27.9	13.4	6.4	14.9	2.9	2.2	2.6	0.58
7	10.1	18.2	22.5	12.6	5.8	12.7	3.1	2.2	2.7	0.6
8	8.8	19.1	20.4	12.8	6.1	11.6	3.1	2.1	2.8	0.62
9	8.8	16.5	19.9	11.7	5.3	11.1	3.2	2.1	2.8	0.64
10	9.4	15.4	18.4	9.7	5.6	10.3	3.4	2.3	2.9	0.66
11	8.9	14	18.6	10	5.7	10.7	3.3	2.3	2.9	0.68
12	8.8	14.4	18.3	10.3	5.9	11.1	**3.5**	**2.4**	**3**	0.7

Time Delta. The overall performance of TD is given in Table 5. The highest value exists using spectral clustering, with number of clusters set to two, with the absolute highest value where the threshold value is five seconds. In general, the lower the number of clusters, the higher the score for spectral clustering and K-Means. For DBSCAN, the higher the value of epsilon, the greater the CH score. However, DBSCAN has much lower scores overall than either spectral clustering or K-Means.

Table 5. Calinski-Harabasz Score Mean for Time Delta.

	Spectral				K-Means				DBSCAN				
clusters	1	2	3	5	1	2	3	5	1	2	3	5	epsilon
2	**33**	**33**	**30**	**36**	11	21	25	22	2	3	5	6	0.5
3	26	26	26	31	13	**26**	19	**25**	2	3	5	6	0.52
4	24	26	26	23	**17**	20	**25**	20	2	3	5	7	0.54
5	24	26	24	22	15	17	21	21	2	4	5	7	0.56
6	23	26	22	21	14	18	20	23	2	4	5	8	0.58
7	20	24	20	21	14	18	21	21	3	4	6	8	0.6
8	21	24	21	22	16	17	20	22	3	5	6	9	0.62
9	21	22	20	19	15	17	22	19	3	5	6	9	0.64
10	21	21	22	20	13	20	22	19	3	5	6	10	0.66
11	19	20	19	16	15	21	17	20	3	5	7	10	0.68
12	19	18	18	19	15	18	23	21	**3**	**6**	**7**	**10**	0.7

Frequency. The performance of FQ is presented in Table 6. FQ overall has much lower scores than FNE or TD. The hour binning value has the highest scores on average, followed by the weekday, then the minute. Lower number of clusters has better performance for hour and weekday, while the minute binning value has higher scores with larger number of clusters. The highest overall value of 7.5 occurs when using spectral clustering with the number of clusters set to four and the binning value of hour. DBSCAN has the best performance for lower values of epsilon, however DBSCAN has lower scores generally than either spectral clustering or k-means.

Table 6. Calinski-Harabasz Score mean for Frequency.

	Spectral			K-Means			DBSCAN			
clusters	hr	mn	wkd	hr	mn	wkd	hr	mn	wkd	epsilon
2	5.7	1.5	1.6	4.0	1.4	1.6	2.5	**1.2**	1.5	0.5
3	7.3	1.4	3.3	3.3	1.2	1.6	**2.6**	**1.2**	1.5	0.52
4	**7.5**	1.7	3.7	**5.3**	1.3	2.1	2.2	**1.2**	1.5	0.54
5	6.4	2.2	**3.8**	5.2	1.4	2.8	2.3	1.2	**1.7**	0.56
6	4.9	1.7	3.5	3.7	1.3	3.1	2.3	1.2	1.7	0.58
7	5.1	1.6	3.2	3.8	1.5	**3.1**	2.3	1.2	1.5	0.6
8	4.7	1.8	2.9	3.9	1.5	3.1	2.3	1.2	1.6	0.62
9	4.1	2.5	2.5	3.5	**1.7**	3.0	2.3	1.2	1.6	0.64
10	4.2	2.7	2.5	3.5	1.6	2.9	2.2	1.2	1.6	0.66
11	3.5	**3.4**	2.5	3.6	1.5	2.7	2.0	1.2	1.6	0.68
12	3.8	2.9	2.3	3.5	1.5	3.1	1.8	1.2	1.6	0.7

To summarize, when considering the CH score, the highest values come from the time delta method using spectral clustering with number of clusters set to two. Within this method, the threshold parameter that achieves the absolute highest score is the 5 seconds value with a score of 36, although the other scores are around 30 as well. The second best values come from the time-based withrounding method of FNE using spectral clustering with number of clusters set to five, where it achieves a score of 30.7. Overall, the k-means performance is average for all three features, which is understandable since this method was chosen as a baseline. The DBSCAN clustering algorithm was unable to produce any valuable results for any of the three features, regardless of the value for the epsilon parameter. This indicates that the density based clustering employed by DBSCAN might not be effective in this scenario. Overall, the CH score shows the effectiveness of *SeReIn* by demonstrating high scores with several different combinations of feature extraction techniques and clustering algorithms.

Silhouette Score

The silhouette score, as discussed in the Experimental Setup, finds the average of how well each sensor fits into its assigned cluster versus how well it would fit into a different cluster. It is important to remember that the silhouette scores can range from [-1,1], with higher values indicating better clustering and negative values indicating misclassified samples. In this section, I will follow the same structure as the Calinski-Harabasz Score section, where I showcase every single feature extraction method in a table with a brief discussion, followed by a summary at the end of this section. Boxplots giving additional details about the information summarized in the tables are given in Appendix B.

Frequent Next Event. FNE is presented in Table 7. The highest scores were in the lower number of clusters, 2 and 3 specifically. Once again, time-based with rounding has the best scores overall, especially when applying the spectral clustering algorithm, and the highest score of 0.26 occurs when number of clusters is two. For the silhouette score as well, DBSCAN seems to perform far worse than spectral clustering and K-means clustering. Several combinations produced negative silhouette scores, which can be indicative of overlapping clusters of misclassified sensors. The only methods that remain strictly positive are the time-based and time-based-rounding methods specifically in the spectral clustering algorithm.

Table 7. Silhouette Score mean for Frequent Next Event.

	Spectral			K-Means			DBSCAN			
clusters	con	tb	tbr	con	tb	tbr	con	tb	tbr	epsilon
2	**0.15**	**0.21**	**0.26**	0.08	**0.04**	**0.16**	**-0.29**	**-0.24**	-0.29	0.5
3	0.05	0.18	0.25	**0.12**	-0.1	0.13	**-0.29**	**-0.24**	-0.29	0.52
4	-0.01	0.11	0.22	0.07	-0.1	0.09	**-0.29**	-0.27	**-0.28**	0.54
5	-0.02	0.06	0.19	0.06	-0.15	0.07	**-0.29**	-0.28	-0.29	0.56
6	0	0.09	0.16	0.06	-0.17	0.05	-0.3	-0.28	-0.29	0.58
7	-0.05	0.08	0.13	-0.01	-0.17	0.02	**-0.29**	-0.28	-0.29	0.6
8	-0.05	0.1	0.11	0.02	-0.19	-0.05	-0.31	-0.32	-0.3	0.62
9	-0.05	0.1	0.09	-0.05	-0.22	-0.04	-0.31	-0.32	-0.3	0.64
10	-0.03	0.06	0.09	-0.09	-0.23	-0.08	-0.3	-0.32	-0.33	0.66
11	-0.02	0.07	0.1	-0.11	-0.25	-0.12	-0.31	-0.33	-0.33	0.68
12	-0.03	0.07	0.13	-0.12	-0.24	-0.12	-0.3	-0.33	-0.32	0.7

Time Delta. TD is presented in Table 8. We see higher values for TD than FNE across all clustering techniques and threshold values. Specifically, the highest value of 0.4 is in the spectral clustering algorithm with the 5 second threshold with number of clusters set to 2. However, the thresholds do not seem to make much of an impact on the score, as the scores in the other thresholds are at most off by 0.1. The scores are highest for lower number of clusters such as 2 and 3 and tend to drop off as the number of clusters increases. In TD there are no negative values for the spectral and k-means clustering methods, only in DBSCAN, indicating that only DBSCAN has misclassified samples.

Table 8. Silhouette Score mean for Time Delta.

	Spectral				K-Means				DBSCAN				
clusters	1	2	3	5	1	2	3	5	1	2	3	5	epsilon
2	**0.3**	**0.3**	**0.3**	**0.4**	0.1	**0.2**	**0.2**	**0.2**	-0.3	-0.2	-0.2	-0.1	0.5
3	0.2	0.2	0.2	0.3	0.1	0.2	0.2	0.2	-0.3	-0.2	-0.2	-0.1	0.52
4	0.2	0.2	0.2	0.2	**0.1**	0.1	0.2	0.1	-0.3	-0.2	-0.2	-0.1	0.54
5	0.2	0.2	0.2	0.2	0.1	0.1	0.1	0.1	-0.3	-0.2	-0.2	-0.1	0.56
6	0.2	0.2	0.1	0.2	0.1	0.1	0.1	0.2	-0.3	-0.2	-0.2	-0.1	0.58
7	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	-0.3	-0.2	-0.1	-0.1	0.6
8	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	-0.2	-0.2	-0.1	0.0	0.62
9	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	-0.2	-0.2	-0.1	0.0	0.64
10	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	-0.2	-0.2	-0.1	**0.0**	0.66
11	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	-0.2	-0.1	-0.1	**0.0**	0.68
12	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	**-0.2**	**-0.1**	**-0.1**	**0.0**	0.7

Frequency. Results for FQ are presented in Table 9. We see very low scores regardless of the clustering method used, indicating a failure of the FQ method. In fact, the majority of the scores are negative, with only a few very low positive numbers. The highest score of 0.09 occurs with spectral clustering with number of clusters set to two and using the hour binning value Similar to FNE and TD, the lower number of clusters have better performance, with the higher number of clusters having extremely negative values. Uniquely, the performance using DBSCAN is actually the highest of all three clustering algorithms. While it is still negative for the most part, it is the least negative of all three algorithms. In addition, the minute component actually performs the best, with it having mostly positive values. It is important to note that these values are still far below the values for other feature extraction methods. Therefore I still consider this feature extraction technique to be relatively poor, even though it has some unique quirks in the results.

Similar to CH score, the overall results for the silhouette score favors TD, then FNE, then FQ features. In addition, we once again see the highest scores in the lowest number of clusters, more specifically when using spectral clustering. In fact, the absolute highest silhouette score of 0.4 uses TD with the 5 second threshold, using spectral clsutering and 2 clusters. DBSCAN has one example of good performance in the FQ feature, but otherwise fails to create good clusters. DBSCAN shows some of the lowest silhouette scores, frequently showing negative numbers, which indicate misclassification. The absolute lowest silhouette score comes from the FQ feature with the minute bin on K-Means clustering with 11 clusters. The silhouette score does showcase the effectiveness of *SeReIn* by demonstrating several ways of achieving highly positive scores utilizing the FNE and TD feature extraction techniques.

Table 9. Silhouette Score mean for Frequency.

	Spectral			K-Means			DBSCAN			
clusters	hr	mn	wkd	hr	mn	wkd	hr	mn	wkd	epsilon
2	**0.09**	**0.06**	**0.05**	**0.08**	**0.07**	**0.07**	-0.15	**0.08**	0.02	0.5
3	0.02	-0.20	-0.08	-0.02	-0.20	-0.15	-0.15	**0.08**	0.02	0.52
4	-0.03	-0.32	-0.17	-0.07	-0.32	-0.24	-0.13	**0.08**	0.01	0.54
5	-0.06	-0.39	-0.20	-0.10	-0.40	-0.28	-0.12	0.07	**0.03**	0.56
6	-0.12	-0.43	-0.24	-0.19	-0.44	-0.31	-0.14	0.07	-0.03	0.58
7	-0.15	-0.47	-0.25	-0.22	-0.45	-0.35	-0.12	0.07	-0.04	0.6
8	-0.20	-0.46	-0.35	-0.25	-0.49	-0.37	-0.13	0.07	-0.04	0.62
9	-0.24	-0.42	-0.39	-0.29	-0.50	-0.40	-0.10	0.07	-0.02	0.64
10	-0.27	-0.43	-0.41	-0.32	-0.51	-0.40	-0.08	0.06	-0.02	0.66
11	-0.30	-0.40	-0.44	-0.33	-0.52	-0.42	**-0.06**	0.06	-0.02	0.68
12	-0.31	-0.40	-0.46	-0.35	-0.50	-0.43	**-0.06**	0.06	-0.01	0.7

Since the silhouette score is actually just an average of the score of every sample, it is uniquely able to calculate scores for every single sample in order to identify if a certain category of sensor systematically lowered the average silhouette score. In aggregate, I counted the worst sensor for every single combination of testbed, clustering algorithm and parameter, and feature extraction method. The results are given in Table 10. We see that the temperature and light sensors are the worst sensor most often, followed by motion sensor then door sensor. This could be due to the fact that the temperature and light sensors both measure continuous

values that can fall within a wide range of possibilities, whereas the motion and door sensors only have two possible values each. In addition, there could be some relationship between the number of events related to each sensor being different for each kind of sensor.

Table 10. Worst Sensor by Category According to Silhouette Score.

Sensor Category	Occurrences of Worst Sensor
Temperature	3175
Light	1384
Motion	805
Door	437

Davies-Bouldin Score

The DB score, as discussed in Experimental Setup, measures how distinct each cluster is from its most similar cluster. It can be any positive value, with lower numbers indicating better clustering. Boxplots giving additional details about the information summarized in these tables are shown in Appendix C.

Frequent Next Event. Results for FNE are presented in Table 11. The lowest scores are generally in found using the spectral clustering method with number of clusters set to two, with the absolute lowest score of 1.24 from the time-based with rounding parameter. In general for these results, the lower the number of clusters or the lower the value of epsilon, the better the scores. Here DBSCAN actually achieves lower scores than k-means very consistently, with even its highest scores being lower than k-means lowest. This contrasts with the previous two evaluation metrics, where DBSCAN always had the worst performance.

Table 11. Davies-Bouldin Score mean for Frequent Next Event.

	Spectral			K-Means			DBSCAN			
clusters	con	tb	tbr	con	tb	tbr	con	tb	tbr	epsilon
2	**1.28**	**1.47**	**1.24**	**3.48**	4.43	2.88	**1.58**	2.14	**1.73**	0.5
3	1.44	1.6	1.37	3.59	4.65	**2.06**	1.6	2.22	1.91	0.52
4	1.6	3.3	1.37	3.83	**2.67**	2.36	1.67	2.18	2	0.54
5	1.86	3.19	1.43	3.49	3.3	2.27	3.17	2.17	2.07	0.56
6	1.84	2.46	1.58	3.63	3.27	2.45	3.61	**2.13**	3	0.58
7	2.47	2.29	1.91	3.75	3.68	2.69	2.14	2.38	2.84	0.6
8	2.19	2.02	1.64	5.8	3.25	3.02	2.19	2.51	2.94	0.62
9	2.86	2.35	1.9	4.08	3.39	5.41	2.21	2.5	2.87	0.64
10	2.24	2.32	1.96	11.89	4.04	4.53	2.21	2.61	2.87	0.66
11	2.1	2.36	2.17	4.53	4.39	5.43	2.33	2.6	2.87	0.68
12	2.98	2.69	2.08	4.25	5.14	4.4	2.6	2.62	2.97	0.7

Time Delta. The results for TD are shown in Table 12. We see slightly lower scores overall for TD than FNEThe lowest scores appear in spectral clustering with 2 clusters, with the absolute lowest of 1.24 using the time-based with rounding increment method. Each threshold value performs about the same. DBSCAN achieves lowers scores than K-Means, however the difference between the two is relatively small. The value of 11.89 for K-Means with number of clusters set to ten and using the constant increment method is an extreme anomaly that makes it difficult to compare these methods.

Table 12. Davies-Bouldin score mean for Time Delta.

	Spectral				K-Means				DBSCAN				
clusters	1	2	3	5	1	2	3	5	1	2	3	5	epsilon
2	**1.0**	**1.1**	**1.1**	**1.0**	2.7	2.3	3.1	2.5	1.5	1.7	1.7	1.8	0.5
3	1.4	1.4	1.3	1.4	2.4	2.0	1.9	2.0	1.6	**1.5**	**1.5**	2.0	0.52
4	1.3	1.2	1.2	1.3	2.6	3.2	2.6	2.4	1.5	1.7	1.5	1.7	0.54
5	1.2	1.2	1.3	1.3	**1.7**	**1.8**	1.8	1.9	1.6	1.7	1.6	1.7	0.56
6	1.4	1.7	1.5	1.4	2.3	2.2	2.0	1.9	1.6	1.7	1.5	1.7	0.58
7	1.4	1.5	1.5	1.6	3.0	2.3	1.9	2.3	1.7	1.7	1.6	1.6	0.6
8	1.5	1.4	1.8	1.7	3.0	2.9	1.7	2.8	1.6	1.7	1.7	**1.6**	0.62
9	1.6	1.6	1.7	2.1	2.2	2.3	1.9	2.9	1.5	1.6	1.8	**1.6**	0.64
10	1.9	1.7	1.7	1.9	2.5	2.3	**1.7**	2.3	**1.5**	1.7	1.8	1.6	0.66
11	2.5	2.1	1.9	2.1	2.3	1.9	1.8	**1.7**	1.5	1.8	1.7	1.6	0.68
12	2.3	2.3	1.7	2.1	2.3	2.3	2.2	2.1	1.7	8.8	1.8	1.7	0.7

Frequency. The results for FQ are shown in Table 13. Like with the CH and SI scores, the FQ feature performs worse than either FNE or TD. Almost all scores using the FQ method are higher than even the highest scores of either FNE or TD. For spectral clustering,the minute bin has the lowest score of 1.98 with number of clusters set to 5, and this is almost comparable to FNE and TD. For the k-means and DBSCAN clustering algorithms, the weekday bin has a wide variety of the lowest scores, but also some of the highest scores. It is difficult to identify any sort of pattern in the performance when comparing across different number of clusters. Overall

this is further indication of the relative failure of the FQ feature to properly identify sensor relationships.

Table 13. Davies-Bouldin score mean for Frequency.

	Spectral			K-Means			DBSCAN			
clusters	hr	mn	wkd	hr	mn	wkd	hr	mn	wkd	epsilon
2	5.95	2.63	3.97	7.63	4.45	**2.76**	4.56	**4.66**	**2.45**	0.5
3	5.43	2.31	5.41	6.9	5.19	3.39	4.84	**4.66**	**2.45**	0.52
4	**4.49**	2.12	4.52	6.72	4.01	4.11	5	**4.66**	2.97	0.54
5	5.53	**1.98**	5.83	8.74	**2.59**	3.18	4.42	4.75	2.66	0.56
6	5.79	3.47	6.44	6.89	4.1	3.94	4.18	4.75	8.62	0.58
7	5.34	2.36	4.93	5.93	3.81	3.63	4.12	4.75	8.27	0.6
8	5.31	2.1	4.41	8.23	4.91	3.56	4.09	4.75	8.06	0.62
9	5.38	2.75	4.93	5.64	3.91	3.4	4.06	5.36	8.01	0.64
10	5.53	3.02	4.41	**5.51**	4.19	3.26	**3.7**	4.78	8.01	0.66
11	5.38	3.08	**3.78**	5.56	4.7	3.9	3.91	4.78	8.01	0.68
12	5.1	7.4	7.3	6.72	3.4	2.93	4.65	4.84	7.94	0.7

Overall, the DB scores are lowest for the time delta method with spectral clustering. Unlike the silhouette score and CH score, DBSCAN show slightly better performance across all methods than K-Means. Deeper analysis in Appendix C shows the existence of some outliers in the DB scores exceeding 100. This large of an outlier heavily influences the average, making it difficult to compare against the other two evaluation metrics.

The CH and SI scores were extremely helpful in identifying the most effective methods for sensor relationship inference. The DB score has some slightly contradictory results and the presence of outliers in the data makes it difficult to justify using this method to evaluate the proposed approach.

Custom Dataset

The previous section dealt with performing objective evaluations on the CASAS dataset. For the custom dataset, the ground truth clustering of sensors is already known from the start. Therefore, I am able to directly compare the cluster results to the ground truth values for each sensor. For simplicity, I utilize only the spectral clustering algorithm. The results of this analysis are summarized in Table 14, where I show the clustering results compared against the sensor names and the raspberry pi devices to which each sensor is attached. Both FNE and TD were able to successfully group sensors according to their related device regardless of the parameters used, which corroborates the findings from the CASAS dataset. This is shown in Figure 9, and since both methods arrive at the same result they are combined into the same figure. The FQ feature, however, produced seemingly random results, and only managed to group the kitchen sensors together. The results for FQ are shown in Figure 10, with numerical results shown in Table 15. Scores Using Frequency on Custom Dataset. This provides even more support to conclusion that this method is unsuitable for sensor relationship inference.

Table 14. Custom Dataset Clustering Results

		FNE	TD	FQ		
Pi #	Sensor ID	All	All	wkd	hr	mn
1	bathroom_door	1	1	4	1	3
1	bathroom_light	1	1	2	2	3
1	bathroom_switch	1	1	3	4	2
2	bedroom_light	2	2	1	3	4
2	bedroom_switch	2	2	2	3	1
3	kitchen_light	3	3	3	3	3
3	kitchen_motion	3	3	3	3	3
4	living_light	4	4	1	1	1
4	living_switch	4	4	3	3	3

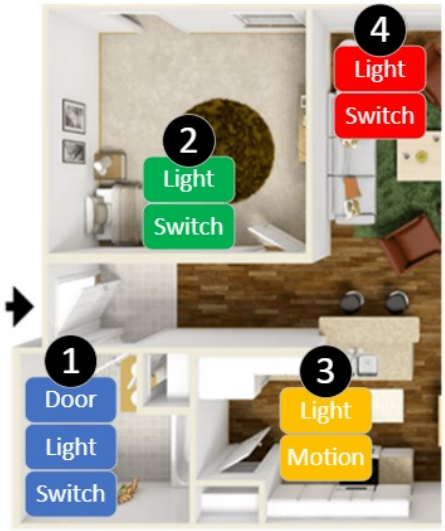


Figure 9: FNE and TD Clustering Results for Custom Dataset.

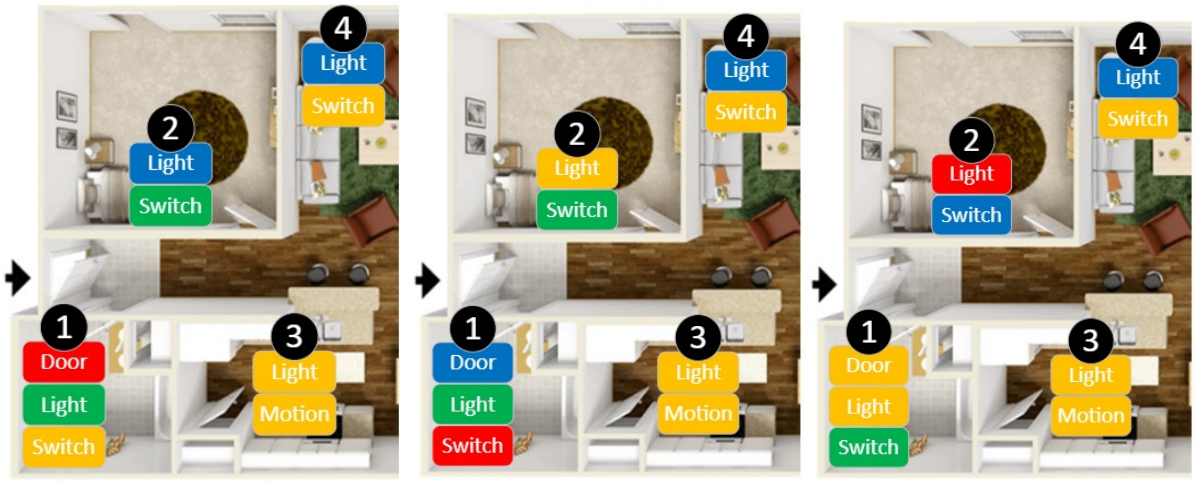


Figure 10: FQ Clustering Results for Custom Dataset using weekday, hour, and minute bins.

Table 15. Scores Using Frequency on Custom Dataset

Scoring Metric	Hour	Minute	Weekday
Calinski-Harabasz Score	0.7	0.8	0.85
Silhouette Score	-0.11	-0.45	-0.41
Davies-Bouldin Score	2.92	1.62	2.29

CONCLUSION

In this thesis, I presented a novel method for identifying spatial relationships between sensors in smart homes. If I consider a smart home as one big system, then this approach will help to determine how sensors are connected to obtain contextual information. It will facilitate the generation of complex operational rules involving multiple sensor triggers, enable automation of smart actions based on events concentrated in sensor groups, and aid in optimizing sensor placements. By using several feature extraction techniques on time series data, *SeReIn* is able to create clusters that describe the inherent relationships between sensors. It achieves CH scores greater than 90 on the CASAS dataset, and arrives at the ground truth clustering for the custom dataset. Overall, the Frequent Next Event and Time Delta features show the most promise for sensor relationship inference, especially when coupled with the Spectral Clustering algorithm. They consistently achieve higher scores than the Frequency feature, and Spectral Clustering outperforms the K-Means and DBSCAN clustering techniques. One of the benefits of *SeReIn* is the low time and space complexity. The feature extraction component runs in O(T) time, where T is the length of the time series, and the results can be saved for later analysis, requiring $O(n^2)$ space to store the adjacency matrix, where n is the number of unique sensors in the smart home. If further analysis is desired after more sensor events have occurred, these saved results can be used in order to reduce the amount of computations for future analysis. Next, the clustering component runs in $O(n^2)$ time, since it is clustering a $n \times n$ matrix. Altogether, *SeReIn* runs in $O(n^2 + T)$ time, which is an improvement over kSVD [16] and deep metric learning [6] approaches.

From the evaluation in the Section Results and Discussion, I am able to determine that overall the TD feature extraction technique performs the best. Across multiple different evaluation metrics, TD, combined with spectral clustering, achieved the best scores, regardless of the value of the threshold parameter. The next best method was time-based with rounding on the FNE feature, again with spectral clustering. I conclude that the FQ feature overall does not capture relevant information for sensor relationship inference. Future work could potentially explore different binning values, but I would recommend instead focusing on expanding upon the two successful feature extraction techniques.

I also determined that the spectral clustering algorithm generally performed the best, especially when combined with the more effective feature extraction techniques. Typically, this was followed by K-Means. When using the DB score, however, DBSCAN actually had slightly better performance in some scenarios, which indicates that there might be future work exploring the interaction between this evaluation technique and clustering algorithm.

In terms of limitations, *SeReIn* does not take into account rare patterns and outliers. Further, all testing was conducted on single-resident smart homes, and the novel features were designed with a single-resident environment in mind. Thus future work might focus on extending the framework developed in thesis to consider dual-resident or multi-resident smart homes. Using the silhouette score, I determined that the most commonly misclassified sensors in the CASAS dataset are the temperature sensors and light sensors. This could be due to the fact that these two kinds of sensors measure continuous variables that can take on a wide range of values while the door and motion sensor can only hold two values each. Future work could potentially include a wider variety of sensors to further identify if certain kinds of sensors

are less effectively grouped. In addition, future work could evaluate the effect of the amount of data on the performance of *SeReIn* to show how quickly it determines effective clusters and also to alleviate the issues caused by rare patterns. Additionally, this could help identify how much historical data should be retained, and how often historical data should be purged. Further, future work could experiment with adding sensors to established deployments to determine how those sensors are assimilated into existing clusters.

REFERENCES

[1] International Data Corporation, "Worldwide Spending on the Internet of Things is Forecast to Surpass $1 Trillion in 2026, According to a New IDC Spending Guide," [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS50936423. [Accessed 30 August 2023].

[2] IoT Analytics, "IoT connections market update—May 2023," [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/. [Accessed 29 August 2023].

[3] D. Cook and S. K. Das, *Smart environments: technology, protocols, and applications*, John Wiley & Sons, 2004.

[4] Amazon.com, "Alexa Smart Home," [Online]. Available: https://www.amazon.com/alexa-smart-home/b?ie=UTF8&node=21442899011. [Accessed 11 October 2023].

[5] Samsung, "Do the SmartThings," [Online]. Available: https://www.samsung.com/us/smartthings/do-the-smartthings/. [Accessed 11 October 2023].

[6] S. Li, D. Hong and H. Wang, "Relation Inference among Sensor Time Series in Smart Buildings with Metric Learning," *AAAI Conference on Artificial Intelligence,* vol. 34, no. 04, pp. 4683-4690, 2020.

[7] E. Weisstein, "Complete Graph," [Online]. Available: https://mathworld.wolfram.com/CompleteGraph.html. [Accessed 01 May 2023].

[8] M. a. K. L. a. L. A. Längkvist, "A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling," *Pattern Recognition Letters,* vol. 42, pp. 11-24, 2014.

[9] P. Dayan, M. Sahani and G. Deback, "Unsupervised Learning," *The MIT Encyclopedia of the Cognitive Sciences,* pp. 857-859, 1999.

[10] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System," *INFOCOM 2000. Conference on Computer Communications. Annual Joint Conference of the IEEE Computer and Communications Societies,* vol. 2, pp. 775-784, 2000.

[11] Q. Yao, F.-Y. Wang, H. Gao, K. Wang and H. Zhao, "Location Estimation in ZigBee Network based on Fingerprinting," *IEEE International Conference on Vehicular Electronics and Safety,* pp. 1-6, 2007.

[12] A. Likhith and M. P. R. S. Kiran, "A Novel Distance Estimation Framework for PDR Based Indoor Localization Using RNNs," *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM),* pp. 1-6, 2023.

[13] A. Nanopoulos, R. Alcock and Y. Manolopoulos, "Feature-based Classification of Time-Series Data," *International Journal of Computer Research,* vol. 10, no. 3, pp. 49-61, 2001.

[14] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," *IEEE International Conference on Computer Vision,* vol. 2, pp. 110-1157, 1999.

[15] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR),* vol. 1, pp. 886-893, 2005.

[16] S. V. Yazdi and A. Douzal-Chouakria, "Time Warp Invariant kSVD: Sparse Coding and Dictionary Learning for Time Series under Time Warp," *Pattern Recognition Letters,* vol. 112, pp. 1-8, 2018.

[17] J. Riley, "Understanding metadata," *Washington DC, United States: National Information Standards Organization,* vol. 23, pp. 7-10, 2017.

[18] A. Schumann, J. Ploennigs and B. Gorman, "Towards Automating the Deployment of Energy Saving Approaches in Buildings," *ACM Conference on Embedded Systems for Energy-Efficient Buildings,* pp. 164-167, 2014.

[19] A. A. Bhattacharya, D. Hong, D. Culler, J. Ortiz, K. Whitehouse and E. Wu, "Automated Metadata Construction to Support Portable Building Applications," *International Conference on Embedded Systems for Energy-Efficient Built Environments,* pp. 3-12, 2015.

[20] M. Pritoni, A. Bhattacharya, D. Culler and M. Modera, "A Method for Discovering Functional Relationships between Building Components from Sensor Data," *International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys),* 2015.

[21] J. Koh, B. Balaji, V. Akhlaghi, Y. Agarwal and R. Gupta, "Quiver: Using Control Perturbations to Increase the Observability of Sensor Data in Smart Buildings," *CoRR,* 2016.

[22] S. Jianbo and J. Malik, "Normalied cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, 2000.

[23] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability,* vol. 1, no. 14, pp. 281-297, 1967.

[24] M. Ester, H.-P. Kriegel, J. sander and X. Xu, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.," *KDD,* vol. 96, no. 34, pp. 226-231, 1996.

[25] D. J. Cook, "Learning Setting-Generalized Activity Models for Smart Spaces," *IEEE Intelligent Systems,* vol. 27, no. 1, p. 32, 2012.

[26] T. Caliński and J. Harabasz, "A Dendrite Method for Cluster Analysis," *Communications in Statistics-theory and Methods,* vol. 3, no. 1, pp. 1-27, 1974.

[27] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics,* vol. 20, pp. 53-65, 1987.

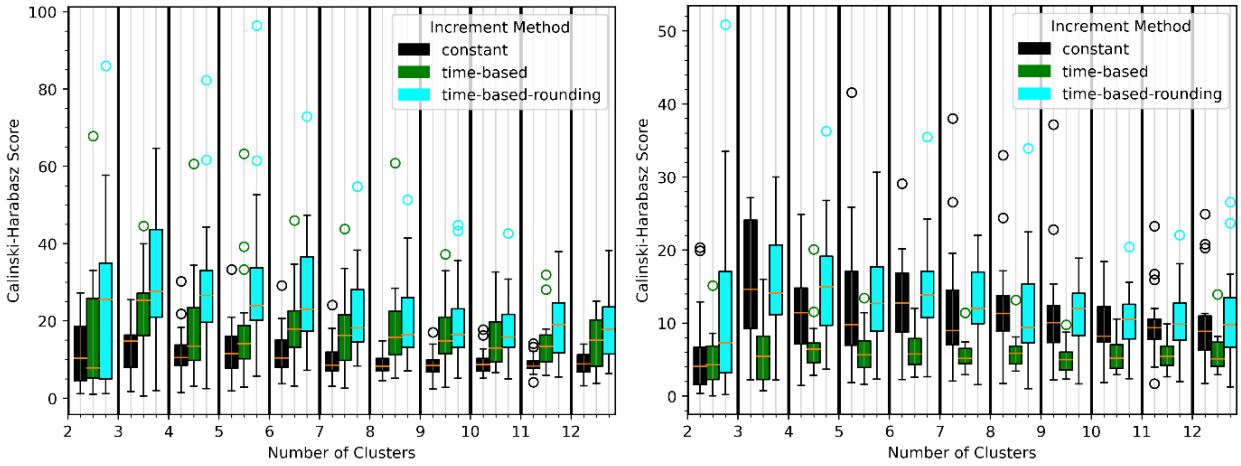
[28] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence,* pp. 224-227, 1979.

[29] G. Lampropoulos, K. Siakas and T. Anastasiadis, "Internet of Things in the Context of Industry 4.0: An Overview," *International Journal of Entrepreneurial Knowledge,* vol. 7, no. 1, 2019.

[30] S. Nack, R. Iqbal and S. Liu, "SeReIn: Smart Home Sensor Relationship Inference," *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM),* pp. 1-6, 2023.

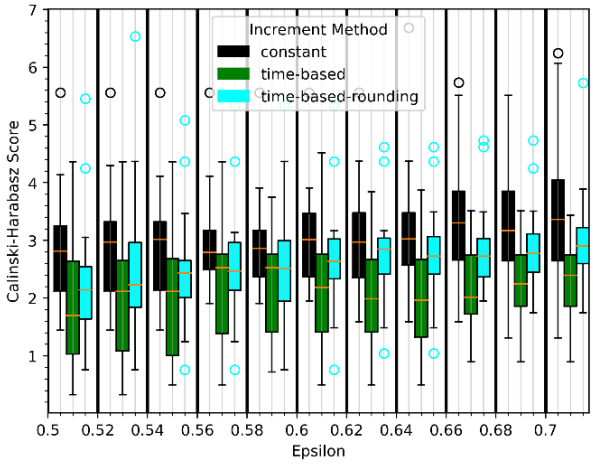
APPENDICES

Appendix A: Calinski-Harabasz Boxplots



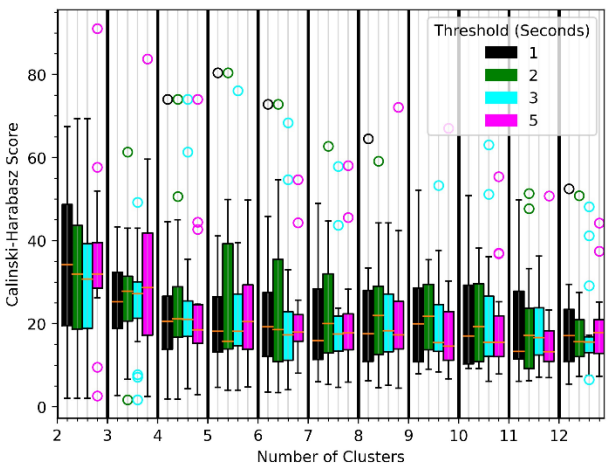
(a) Spectral Clustering CH Score

(b) K-Means Clustering CH Score

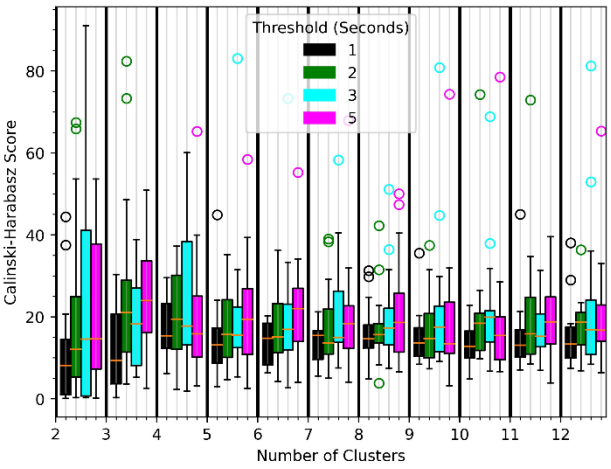


(c) DBSCAN CH Score

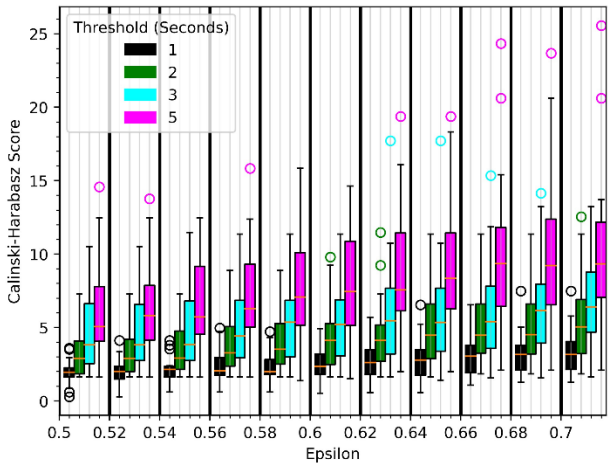
Appendix A-1. Frequent Next Event Feature Calinski-Harabasz Score Boxplots.



(a) Spectral Clustering CH Score

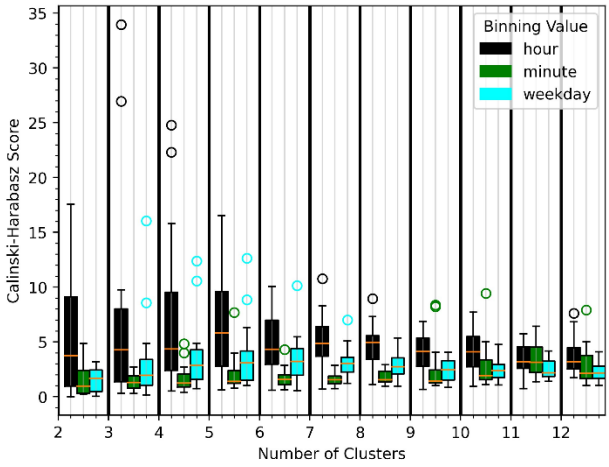


(b) K-Means Clustering CH Score

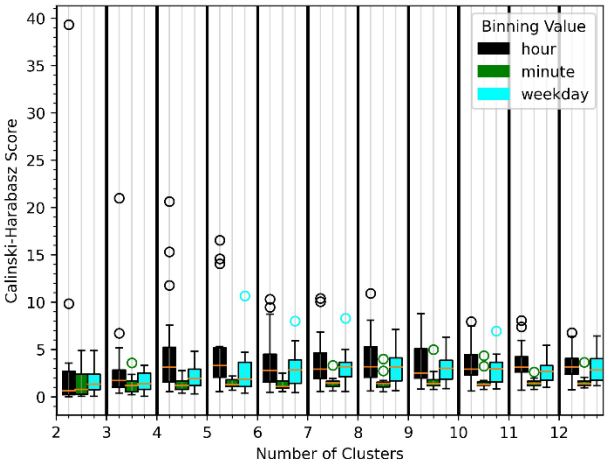


(c) DBSCAN CH Score

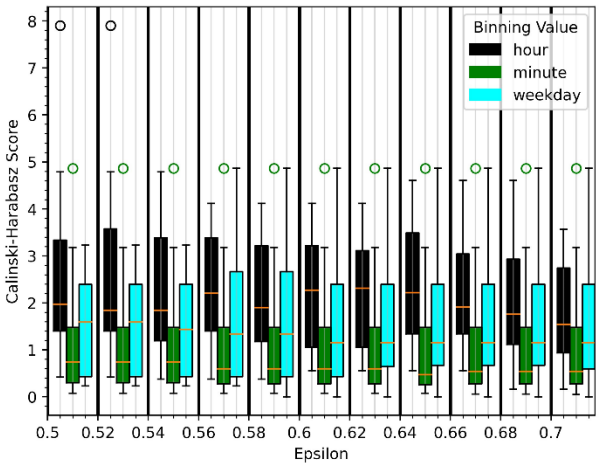
Appendix A-2. Time Delta Feature Calinski-Harabasz Score Boxplots.



(a) Spectral Clustering CH Score



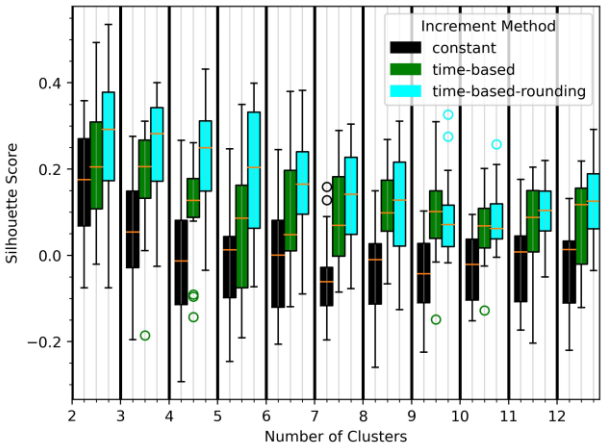
(b) K-Means Clustering CH Score



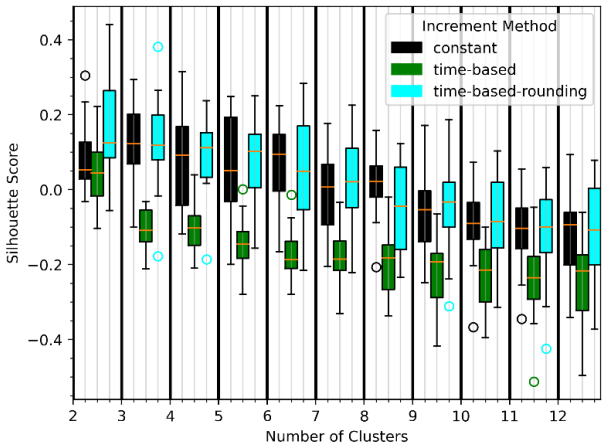
(c) DBSCAN CH Score

Appendix A-3. Frequency Feature Calinski-Harabasz Score Boxplots.

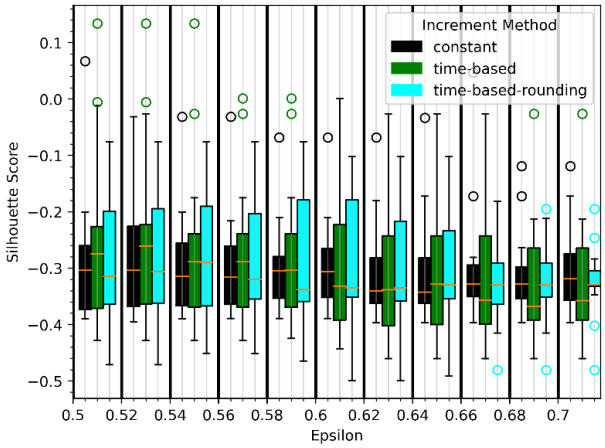
Appendix B: Silhouette Score Boxplots



(a) Spectral Clustering Silhouette Score

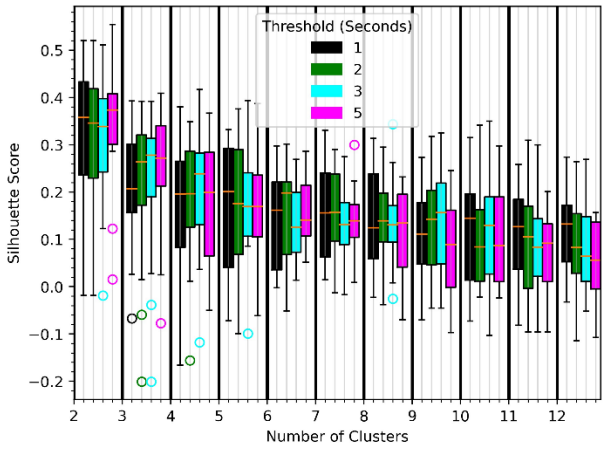


(b) K-Means Clustering Silhouette Score

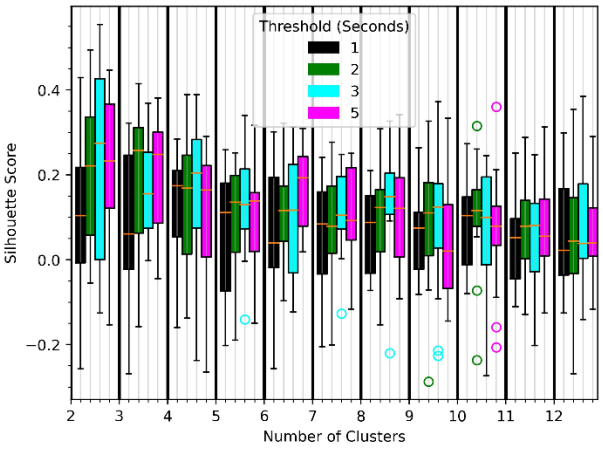


(c) DBSCAN Silhouette Score

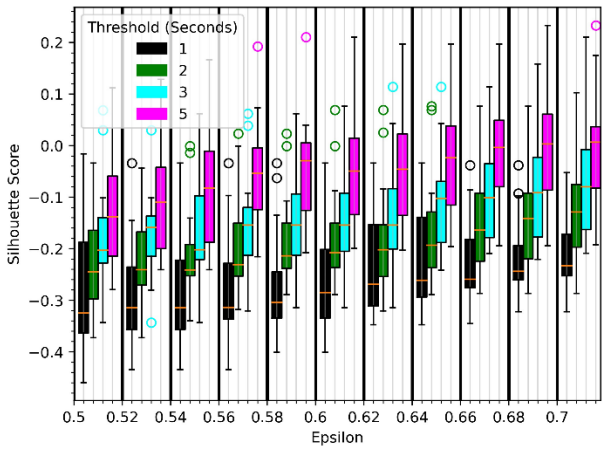
Appendix B-1. Frequent Next Event Feature Silhouette Score Boxplots.



(a) Spectral Clustering Silhouette Score

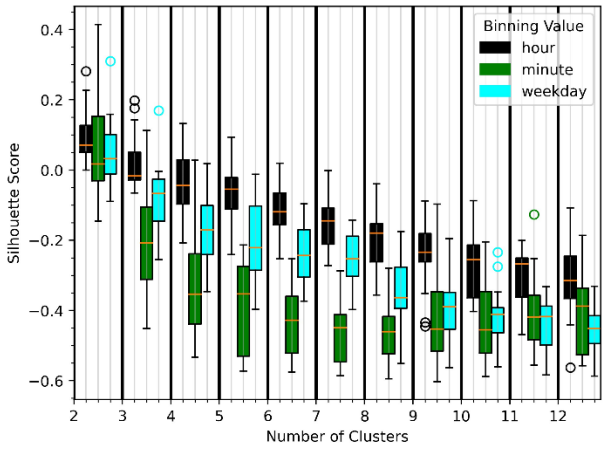


(b) K-Means Clustering Silhouette Score

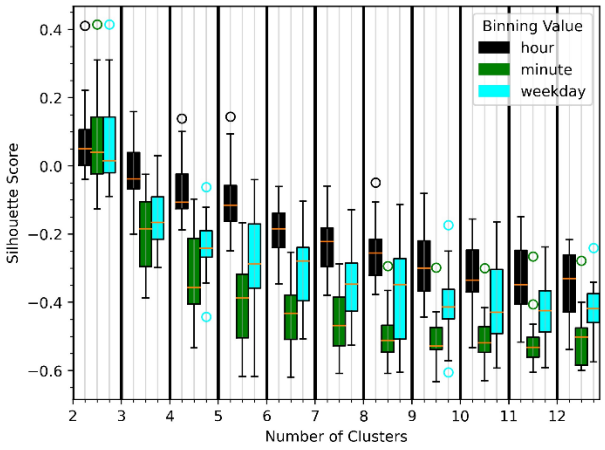


(c) DBSCAN Silhouette Score

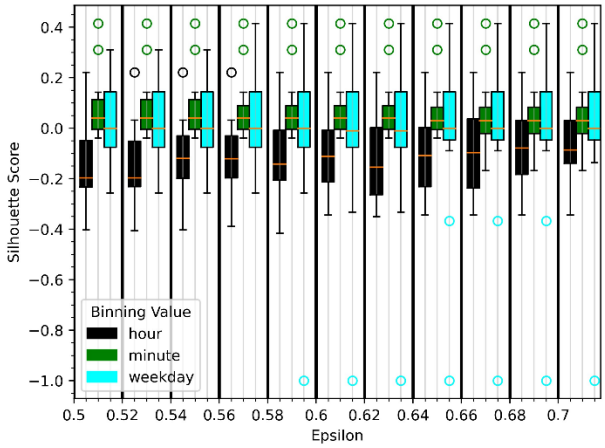
Appendix B-2. Time Delta Feature Silhouette Score Boxplots.



(a) Spectral Clustering Silhouette Score



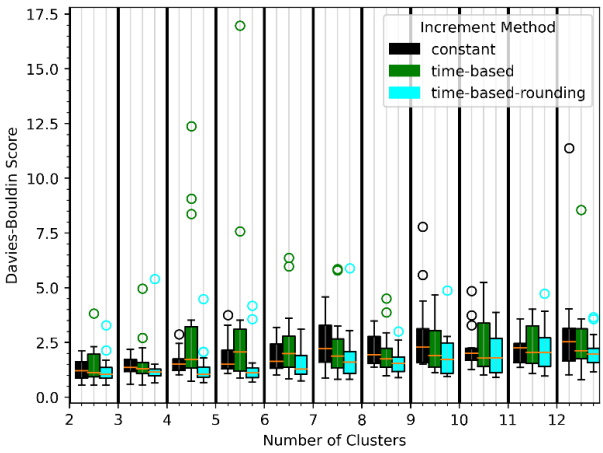
(b) K-Means Clustering Silhouette Score



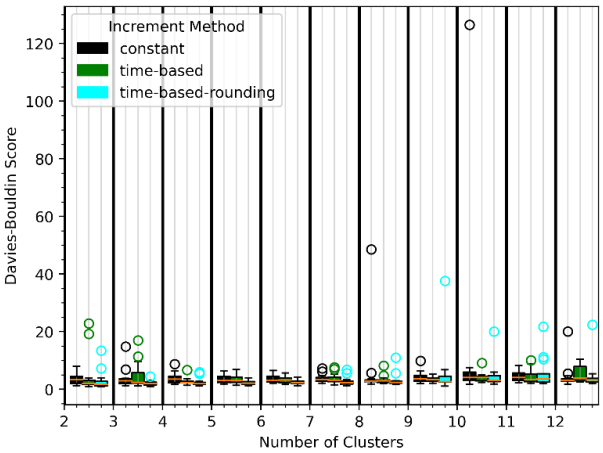
(c) DBSCAN Silhouette Score

Appendix B-3. Frequency Feature Silhouette Score Boxplots.

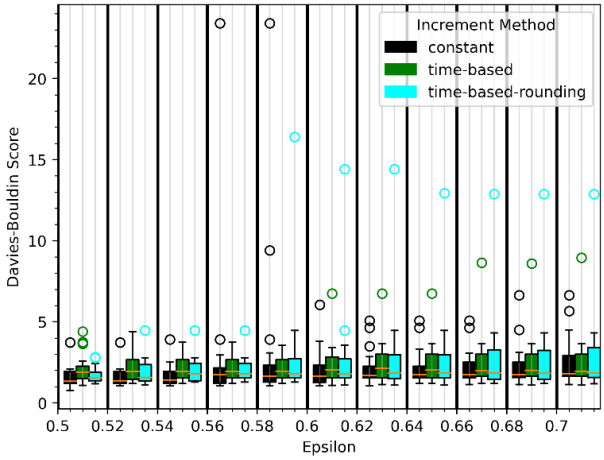
Appendix C: Davies-Bouldin Boxplots



(a) Spectral Clustering DB Score

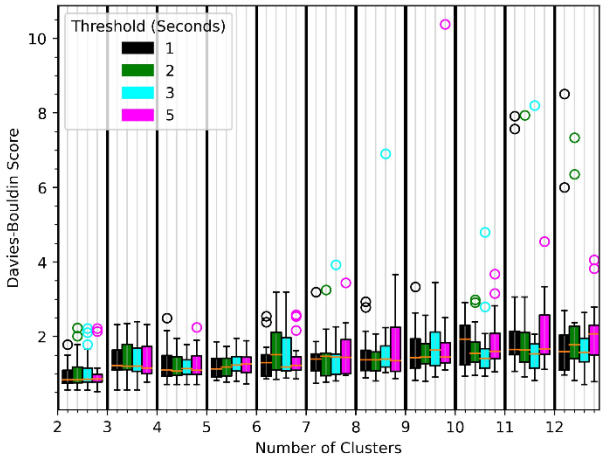


(b) K-Means Clustering DB Score

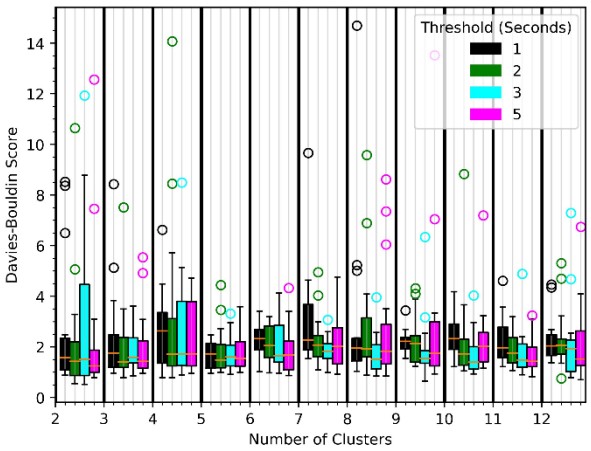


(c) DBSCAN DB Score

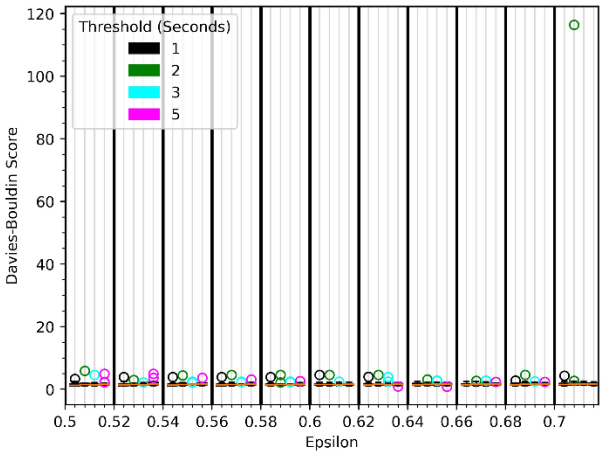
Appendix C-1. Frequent Next Event Feature Davies-Bouldin Score Boxplots.



(a) Spectral Clustering DB Score

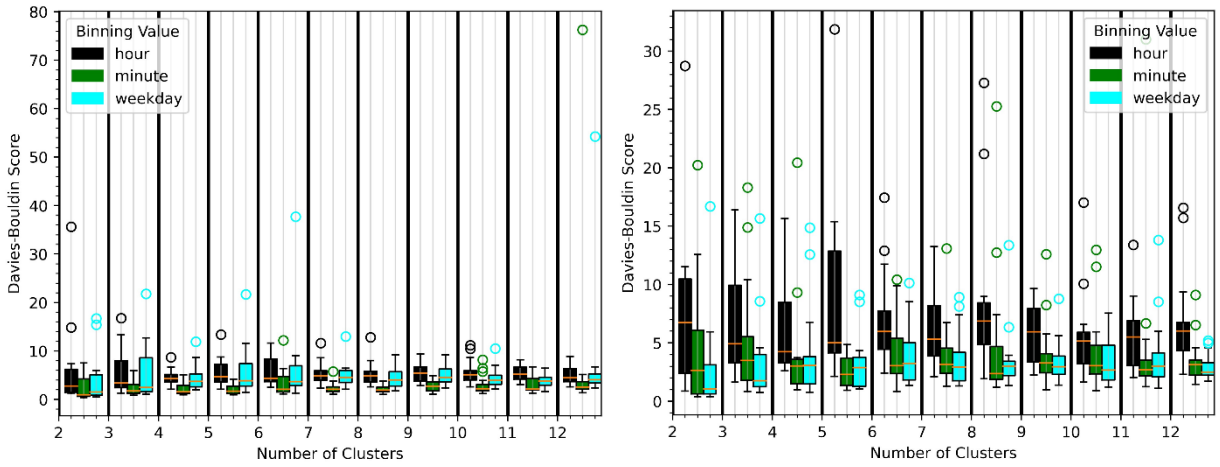


(b) K-Means Clustering DB Score

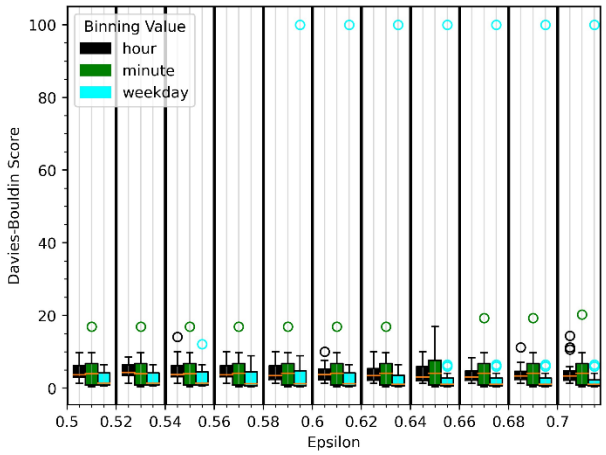


(c) DBSCAN DB Score

Appendix C-2. Time Delta Feature Davies-Bouldin Score Boxplots.



(a) Spectral Clustering DB Score (b) K-Means Clustering DB Score



(c) DBSCAN DB Score

Appendix C-3. Frequency Feature Davies-Bouldin Score Boxplots.