# A Study on Compression-based Sequential Prediction Methods for Occupancy Prediction in Smart Homes

Sreerama Krishna Sama and Mahshid Rahnamay-Naeini
Department of Computer Science, Texas Tech University, Lubbock, Texas, USA
sreeram.sama@ttu.edu, mahshid.naeini@ttu.edu

***Abstract*—This paper studies the occupancy and movement prediction of residents in the smart home based on a compression-based sequential prediction approach and discusses home automation applications that can benefit from such predictions. The prediction approach studied here is based on the Active LeZi algorithm, which is a compression-based approach and uses an order-k Markov model. The effects of the order of the memory in the Markov model on the prediction accuracy have been investigated and it has been suggested that limiting the memory of the Markov chain model in the Active LeZi algorithm can improve the performance of the prediction. Moreover, the effects of the interrupted patterns and the temporal information of the patterns have been added to this prediction model. A small-scale smart home testbed using motion detector sensors and a central microcontroller based on Arduino technology along with synthesized data have been used to study the performance of the prediction models.***

***Keywords—smart homes, sequential prediction models, occupancy prediction, Active LeZi, Markov property***

## I. INTRODUCTION

The intelligence built into the smart homes through the large deployment of sensing, computing and communication elements along with pattern recognition, prediction and adaptive control algorithms introduce new opportunities for better smart home solutions. In particular, by collecting information about residents' activity pattern, resource utilization behavior, comfort level, state of online and stored resources, many researchers have focused on developing activity pattern recognition and prediction techniques, which can enable adaptive algorithms to respond to the dynamic residents' demands.

In this paper, we study the occupancy and movement prediction of residents in the smart home based on a sequential prediction approach using the Active LeZi algorithm [1]. The Active LeZi algorithm is founded on an information theoretic approach and a data compression algorithm, which uses an order-*k* Markov model. This algorithm has been introduced in MavHome project [1] and has been used for mobility prediction in smart home applications. We have specifically evaluated the effects of the order of the memory in the Markov model on the prediction accuracy in this algorithm. Based on this study, we will discuss that limiting the memory of the Markov model in the Active LeZi algorithm can improve the performance of the prediction. In addition, the effects of the interrupted patterns and the temporal information of the patterns have been added to the prediction model. Considering interrupted patterns is specifically important since such patterns are common in smart home applications. Moreover, the inhabitants in a smart home tend to repeat certain patterns or activities throughout the day and the likelihood of activities changes depending on the time of the day. Capturing the temporal information of the observed patterns in the prediction model enhances the performance of the prediction and provide more accurate and time-aware predictions.

In our studies, we use a small-scale smart home testbed using motion detector sensors and a central microcontroller based on Arduino technology along with simulations to evaluate the performance of the prediction algorithms. The small-scale testbed allows us to collect movement data faster and in a supervised manner and serves as an education platform for students to experiment and study smart home technologies. Finally, we discuss smart home applications and services such as energy management and home automation that can benefit from the predictions provided by the models under study.

The rest of the paper is organized as following. In Section II, we present a review of smart home application areas and prediction models that enable such applications. We also briefly review existing smart home projects. Section III, briefly reviews the Active LeZi algorithm with examples. The proposed studies and extensions on Active LeZi have been discussed in Section IV. The overview of the testbed and the experimental results for the proposed studies and extensions are presented in Section V. Section VI describes examples of smart home applications that can benefit from the sequential prediction models. Finally our conclusions are presented in Section VII.

## II. RELATED WORK

In this section, we review related work to the current paper in two main categories: (1) occupancy and activity prediction, and (2) smart home applications and adaptive algorithms.

### A. *Occupancy, Mobility and Activity Prediction*

The occupancy, mobility and activity prediction for smart home applications have been extensively studied in the literature in the past decade. Here, we briefly review examples of such efforts.

The classification-based techniques for activity prediction in smart homes have been studied for instance in [2, 3, 4]. Specifically, in [2] an approach based on neural networks, in [3] a decision tree classifier and in [4] an approach based on support vector machines have been used for activity prediction in smart homes.

Another category of approaches for activity prediction are based on sequential prediction models. An example of such model is, the hidden Markov Model (HMM), which has been studied in [5, 6, 7] for activity recognition and prediction in smart homes. In general, HMM provides flexible structure to model complex sequential data for prediction purposes; however, successful training of HMM model requires large datasets [8].

The approaches that are closely related to this paper are the sequential prediction models, which are based on compression algorithms. Hence, we will discuss such models in more details next. Lossless compression algorithms along with prediction by pattern matching (PPM) family algorithms [9] have been used for prediction purposes in the past. Specifically, LZ78 (Lempel-Ziv compression algorithm) [10] is one of the most popular lossless compression algorithms, which is also used as the basis for many commercial applications of compression, that has been used as the basis of some prediction algorithms. This algorithm is a dictionary-based algorithm, which parses a string sequence of events into substrings that form the set of phrases used for compression. One of the approaches based on LZ78 is LeZi Update algorithm [11], which keeps track of all possible contexts within a given phrase of input symbols while parsing the sequence. However, it does not consider the information lost across phrase boundaries, and have a low convergence rate. LeZi Update algorithm was first proposed for mobility tracking in wireless networks. In another approach called Active LeZi algorithm [1], a sliding window concept has been introduced to address the issue of slow convergence with the prediction algorithms based on the original LZ78 and LeZi Update. Active LeZi algorithm [1] builds an order-*k* Markov model and uses a sliding window, where the size of window is the size of longest phrase seen so far. This window is also used to capture information lost on the phrase boundaries. Other variants of Active LeZi have also been proposed, for example, SPEED algorithm in [12]. The study in this paper is focused on Active LeZi algorithm and thus we discuss this algorithm in detail later in this paper.

### *B. Smart Home Application Areas*

Here, we briefly review some smart-home application areas including resource management and automation, security and health-care applications.

Resource management applications and home automation are mainly concerned about automation and management of critical resources in the smart home. Specifically, key resources in smart home environments include energy (e.g., electricity, gas) and water. Efficient management of such resources are essential for making more sustainable and cost efficient smart homes [13]. Analyzing resource demands of the residents, predicting demands and proposing new algorithms for improving the resource utilization in the smart homes are examples of research problems related to this application. Examples of such research efforts are automation and optimization of the heating/cooling systems [14], lighting systems [15], automation and optimization of water resources [16] for smart homes. Other examples include managing deployed renewable energy resources (solar and storage energy resources) [17, 18] to help reducing the energy cost of smart homes and incorporating weather prediction in resource management of smart homes [19].

Security is another important service that the smart home can offer to its residents [20], while privacy is an issue closely related to the security and has been studied in several research efforts (e.g., [21, 22, 23]).

Health-care and elderly-care are other important examples of smart home services that have been studied extensively, for example, in [24, 25, 26]. Detecting falls, health monitoring and directed medication are a few examples of health-care and elderly-care services.

All these application areas rely on activity and occupancy prediction of residents to effectively respond to their demands.

## III. PREDICTION MODEL BASED ON ACTIVE LEZI

In this section, we review the Active LeZi algorithm. The idea of this approach (and generally LZ78 based approaches) is to break up the sequence of symbols (recorded sequence of activities in time) and build a trie, which allows calculation of the probability of next symbols (activity) based on the information (i.e., the frequency of symbols) stored in the trie. The pseudocode of Active LeZi, as was originally presented in [1], is shown in Figure . This algorithm builds and updates a trie based on the input symbols with their respective frequencies as shown in Figure . The depth of the trie at each time will be the maximum length of phrases that are parsed until that time. After the trie is built, the window consisting of the last phrase is used to generate prediction of the symbol that is most likely to occur. We explain the algorithm with an example.

```
initialize dictionary := null
initialize phrase w := null
initialize window := null
initialize Max_LZ_length = 0
loop
    wait for next symbol v
    if ((w.v) in dictionary):
       w := w.v
    else
       add (w.v) to dictionary
       update Max_LZ_length if necessary
       w := null
    endif

    add v to window
    if (length(window) > Max_LZ_length)
       delete window[0]
    endif
    Update frequencies of all possible
          contexts within window that
       includes v
forever
```

*Figure 1: Active LeZi pseudo code, adopted from* [1].

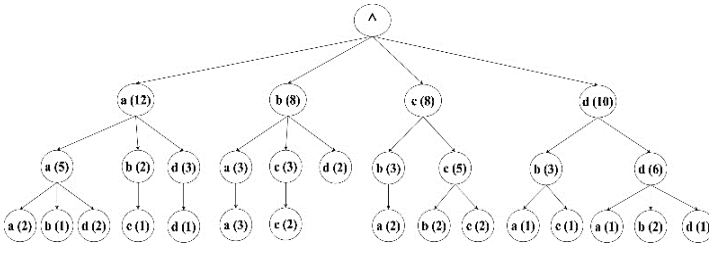


*Figure 2: Trie formed by Active LeZi parsing of the sequence 'aaddabddbccbdddbcccbaaaddbaaabcccbaad'*

Consider the sequence of input symbols '*aaddabddbccbdddbcccbaaaddbaaabcccbaad*'. The algorithm yields a dictionary with contexts according to the trie shown in Fig. 2 and built using the above algorithm.

The Active LeZi algorithm forms an order-*k* Markov model, where *k* is the length of longest phrase seen so far. The Active LeZi algorithm includes more information about the contexts in

the sequence compared to LZ78 due to considering the order-*k* memory in the model using the sliding window.

For an example on calculating the likelihood of various symbols, consider the end of the training sequence above, i.e., *'aad'*, which is also the window to be used for predicting the next symbol. We use the suffixes of this window to calculate probabilities of the next symbol (i.e., the possible contexts to be considered are *'ad'*, *'a'* and *'null'*). A prediction by pattern matching (PPM) algorithm is then used to calculate probabilities [12] as following. The algorithm calculates probability of a phrase by successively reducing the length of the phrase and measures the probability distribution according to the length of the phrase and the relative weights (frequency) of events stored in the trie. The equation for calculating the probability is as following:

$$P_k(\varphi) = p_k(\varphi) + e_k(\varphi).P_{k-1}(\varphi), \quad (1)$$

where $\varphi$ is the symbol to predict its probability of occurrence, $P_k(\varphi)$ is the final probability of the symbol, $p_k(\varphi)$ is the probability of seeing the symbol after the phrase of length *k*, $e_k(\varphi)$ is the escape probability of symbol $\varphi$. The probability of escaping to lower levels is the probability of having 'null' at that level. This equation allows us to recursively calculate the probability of symbols as we move toward lower order contexts in the trie. Suppose we want to estimate the probability of occurrence of event *'b'* after the training sequence seeing above. We start from the highest order, which is *'ad'* in this example. In the trie, we can see that *'b'* does not occur after context *'ad'*. Therefore, the probability of *'b'* at this level is 0/3. Next, we calculate the probability of *'b'* occurring after the context *'d'*, by escaping to the lower level with a probability of 2/3. At this level, *'b'* appears two times out of twelve times that *'a'* appears. Thus, the probability of *'b'* occurring after context *'a'* is 2/12. The probability of null phrase appears two times out of twelve times that *'a'* appears. Thus, after escaping to the lower level with a probability of 2/12, *'b'* appears eight times out of thirty-eight possible phrases. Thus, we have a probability of 8/38 for the latter. The blended probability of the occurrence of the phrase *'b'* according to recursive use of equation (1) is thus

$$\frac{0}{3} + \frac{2}{3}\left\{\frac{2}{12} + \frac{2}{12}\left\{\frac{8}{38}\right\}\right\} = 0.1345.$$

As the phrase is made up of only one symbol, the whole probability is assigned to it. If it has more symbols, the probability is distributed according to their relative frequencies.

## IV. Studies and Extensions on Active LeZi Algorithm

In this section, we introduce the studies and extensions proposed on the Active LeZi algorithm to improve the prediction accuracy.

### A. *Active Lezi with Constrained Memory*

Active LeZi algorithm is an incremental sequential parsing algorithm that uses Markov property to predict the next symbol. In this paper, we study the impact of memory in the Markov model on the accuracy of the Active LeZi algorithm. As shown in Section III, the Active LeZi algorithm considers a variable-length sliding window that can capture information across phrase boundaries and builds an order-*k* Markov model, which lead to converging faster to better predictability. In Active LeZi algorithm, as the sequence grows over time, the patterns recorded get larger and thus, the window size increases. According to the Markov property, the probability of the present event depends only on the immediate previous event. However, in this algorithm, as the window grows larger, it includes more history and memory in the model, thus, it starts to lose the Markov property and thus the prediction accuracy decreases. As such, we propose limiting the growing size of the sliding window in the Active LeZi algorithm to prevent large memory in the model. We specifically define a window-size threshold above which we reset the window size to zero. This approach will reduce the depth of the trie, i.e., the length of patterns, and increase the breadth of the tree, i.e., the tree will contain more patterns. In Section V, we illustrate the effect of memory on the prediction performance. We have identified the window-size threshold that lead to better predictions for the Active LeZi algorithm using extensive experiments.

### B. *Capturing Interrupted Patterns*

To better model real-world inhabitants' behavior, we need to consider patterns that imitate natural inhabitant behavior, for example, interruptions in a pattern. For example, consider the event sequence where an inhabitant moves from the bedroom ('*a*') to the kitchen ('*b*'), and then to the living room ('*c*') in a typical daily activity, recording a pattern of '*abc*'. On a particular day, the inhabitant might forget to, for example, take an item from the bedroom and realizes it after going to the kitchen. He goes back to the bedroom and then continues his typical sequence. This records a new pattern '*abac*'. In reality, this is the same old pattern and can happen often due to the natural human behavior, thus, it is important to count it also toward the '*abc*' context. We capture such behavior by considering interrupted patterns within a pattern that has been recorded previously. For example, in the pattern '*abac*', this approach considers patterns including '*aac*', '*abc*', and '*aba*', and check if any of these patterns are already exist or has been recorded before. If the pattern exists then the algorithm increases the frequency of the pattern. In this case, this approach increases the frequency of the pattern '*abc*' and also records the new pattern '*abac*'. Thus, the existing patterns are recognized and new patterns are added to the trie, which eventually helps in better predictions. We show the improvement of prediction for this extension in Section V.

### C. *Time-aware Prediction*

Residents tend to perform certain patterns depending on the time of the day, for example, cooking. These events may also be repeated during the day but the events that succeed these activities may vary depending on the time of the day. For example, the inhabitant might study after breakfast and sleep after dinner. Therefore, the prediction of events should also depend on the time of the day. However, the existing Active LeZi algorithm does not consider time of the events.

We propose an approach for capturing the temporal information of events in the model and name it "*Time-aware Active LeZi algorithm*". This is done by pairing the frequency attribute (termed the total frequency) with a frequency vector of the form [$f_{t1}$, $f_{(t1, t2)}$, $f_{t2}$, $f_{(t2, t3)}$, $f_{t3}$ …] for every symbol in the trie, where $f_{t1}$ is the frequency of the symbol at time period *t1*, and $f_{(t1,t2)}$ is the frequency of the symbol at the boundary of the time intervals *t1* and *t2*. These frequencies in the frequency vector, along with the total frequency are updated when the trie is being built. The frequency element in the frequency vector to be updated is determined by the time period in the window. Consider this example for demonstration. The input sequence along with

their respective time periods is of the form '*a(M)a(M)b(M)c(M)a(A)d(A)a(A)b(A)c(E)d(E)b(E)a(E)b(N)c(N)b(N)d(N)*', where *M* stands for morning, *A* for afternoon, *E* for evening, and *N* for night. Thus, the frequency vector is of the form [$f_m$, $f_{(m,a)}$, $f_a$, $f_{(a,e)}$, $f_e$, $f_{(e,n)}$, $f_n$, $f_{(n,m)}$]. Consider a case where the window has symbols *a(M)a(M)b(M)*. As all the symbols are in the time period *M*, the frequency count of element $f_m$ in the frequency vector is incremented for each symbol in the window. Consider another case where the window has symbols *c(M)a(A)d(A)*. The window includes symbols of various time periods and therefore, it is on the boundary of *M* and *A*. Thus, the frequency count of the element $f_{(m,a)}$ in the frequency vector is incremented for each symbol in the window. This approach is continued for all symbols in the input stream to build the trie with additional time frequency information. The boundaries enables us to capture the events that potentially belong to either or both of the time intervals.

To calculate the probabilities based on the trie, note that the next symbol should be occurring in the same time period as the last observed symbol or in the succeeding time period. Thus, the equation to calculate the probability of a phrase will be

$$P_k(\varphi) = \sum_{i=u,v}(p_k(\varphi))_i + \sum_{i=u,v} e_k\,(\varphi).(p_{k-1}(\varphi))_i\,, \quad (2)$$

where *u* is the current time period, *v* is the succeeding time period, $(p_k(\varphi))_i$ is the probability of seeing the symbol after the phrase of length *k* in the time period *i*. Hence, the new probability includes the frequency of a symbol in time period *u* (the same time period), and in time period *v* (the succeeding time period). For example, consider the case when the window is *a(M)a(M)b(M)*. The subsequent symbol can be either in time period *M* or *A* due to the order of the time periods in a day. Thus, after reading the new symbol (for example, say '*x*'), the window might be *a(M)b(M)x(M)* or *a(M)b(M)x(A)*. To include both of these possibilities, the probability of the symbol $p_k(\varphi)$ should be the sum of probabilities at time periods $u=t_m$, and $v=t_{(m,a)}$.

In order to use the probability values from both of the total trie as well as time-aware frequencies, we propose a weighted average of the two using a control parameter '$\alpha$' as following

$$(P_{result})_i = \alpha\,.(P)_i + (1-\alpha)\,.(P_t)_i, \quad (3)$$

where $i$ = *a*, *b*,…., are the unique symbols in the training sequence, $(P_{result})_i$ is the total probability of the symbol '$i$', '$\alpha$' is the weight parameter, $(P)_i$ is the probability of the symbol '$i$' obtained using the total trie from Active LeZi algorithm, and $(P_t)_i$ is the probability of the symbol '$i$' obtained from the time-aware frequency information. We show the results of our experiment for the proposed algorithm in Section V.

### D. *Complexity Analysis of the Prediction Model*

Here, we briefly discuss the complexity of the Active LeZi with constrained memory as well as the algorithm with interrupted pattern. The analysis is the simple extension of the Active LeZi complexity analysis as presented in [1]. The worst possible training sequence for Active LeZi rapidly increases the phrase length so that new nodes are added to the trie. As suggested in [1], such a sequence can be represented by S = $x_1$ $x_1x_2$ $x_1x_2x_3$ $x_1x_2x_3$….$x_k$, where the length of the sequence is $n = k(k+1)/2$. Similarly, in our proposed extension for Active LeZi algorithm with fixed window size, this sequence can be represented as S = $x_1$ $x_1x_2$ $x_1x_2x_3$ $x_1x_2x_3$….$x_m$ $x_2x_3$….$x_{m+1}$ $x_3$….$x_{m+2}$… where *m* is the maximum allowed phrase length to limit the order of the memory of the Markov chain model. The length of this sequence is $n = m(m+1)/2 + r.m$, where *r* captures the effect of sliding window on the remaining data points after reaching the maximum window size and grows with the size of input, *n*. In the worst case, each phrase adds a new node to the trie. Creating a new node or updating the frequency of an existing node requires parsing of the phrase, which depends on patterns or phrases found in the sequence. Thus, the time complexity based on the number of contexts added per window and number of sliding windows will be $O(m^2) + O(n.m)$, which for a fixed small window size leads to the time complexity of $O(n)$.

Next, we analyze the time complexity of the algorithm with the extension that considers interrupted patterns but not fixed window size in the Active LeZi algorithm. As discussed in the earlier section, existing patterns in the window are recognized only if they have minor interruptions (one symbol difference). In the worst case scenario, *k* patterns can be recognized, where *k* is the length of the window. So, the trie parses $O(k^2)$ nodes while creating new nodes or if it found an existing context then it will update the frequency of existing nodes. Therefore, the number of nodes parsed in the trie by the time the model attains order *k* is $O(k^3)$ and the number of nodes parsed in the trie while recognizing interrupted patterns is $O(k^4) = O(n^{4/2})$ because $k = O(\sqrt{n})$ and thus the time complexity of the algorithm with the interrupted patterns is $O(n^2)$. Note that the time complexity for interrupted patterns and fixed window size is simply the complexity of the constrained memory approach with a scalar coefficient and thus is $O(n)$ as we have *m* extra operation per window, i.e., $O(m^3) + O(n.m^2)$. Based on similar analysis, the time complexity of the time-aware Active LeZi is the same as the original Active LeZi and due to space limitations, we are not discussing it here.

## V. EXPERIMENTS AND RESULTS

In this section, we present our experiment results for the discussed algorithms. The prediction accuracy rate in these experiments is computed as following:

$$predict.acc\ \% = \frac{no.\ of\ hits}{(no.\ of\ symbols\ in\ training\ sequence)} \times 100, \quad (4)$$

where, a hit is counted when the symbol predicted by the prediction model is the same as the symbol that occurs next in the sequence, indicating that the prediction model predicted the next symbol correctly. Here, we start by reviewing the smart home testbed and the dataset used for these experiments and then present the results.

### A. *Smart Home Testbed and Data Collection*

Many research teams have developed smart home models to learn, and study various aspects of smart homes. Examples of the academic smart-home testbeds are Adaptive House [27], MavHome [28], GatorTech Smart House [29], Aware Home [30]. In contrast to these testbeds, in this research, we implemented a small-scale, smart-home testbed using Arduino microcontroller board [31] and sensors for collecting occupancy information. The architecture of this testbed consist of the following layers. The

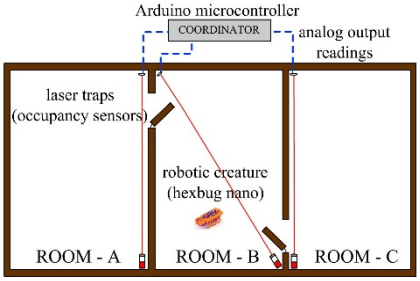


*Figure 3: Floor plan and sensor layout*

'*Action layer*' suggests actions for the smart home based on the information received from the layers below. This layer will be discussed in Section VI. The '*Knowledge layer*' gathers and stores information on inhabitant's occupancy and movement pattern and then uses this information to generate predictions based on the prediction algorithms. The '*Communication layer*' is responsible to collect data from the physical layer and send control signals to the physical layer. The '*Physical layer*' consists of sensors to sense the occupancy and movement patterns, actuators to implement actions and the Arduino microcontroller to coordinate and orchestrate the physical layer by managing the operations of sending and receiving data from these devices.

The layout of the smart-home testbed is illustrated in **Error! Reference source not found.**3. The smart home consists of three rooms and each room has a door, which enables us to supervise the movement of the inhabitant. We use a micro robotic creature called Hexbug Nano [32] to simulate inhabitants. It is powered by a tiny motor and has angular legs that allows it to navigate on a hard surface.

For the purpose of evaluating the performance of the proposed approach in addition to the data collected from the test-bed, we generate a simulated dataset. This is done by pre-defining patterns with varying temporal information on day-to-day basis by using a range of randomness defined for each symbol. For example, consider the pattern '*abbcddabc*' that replicates a realistic inhabitant pattern with each symbol representing a location in the home (e.g., specific room in the house). To indicate the inhabitant occupancy in a room for a certain duration of time, each symbol can have a range of duration, for example, '*a*(*1-3*)*b*(*2-5*)' meaning that the resident may stay in room '*a*' for 1 to 3 time slot duration and in room '*b*' for 2 to 5 time slot duration. The algorithm generates random patterns based on the duration range associated with each symbol as well as small random variations in the order of symbols.

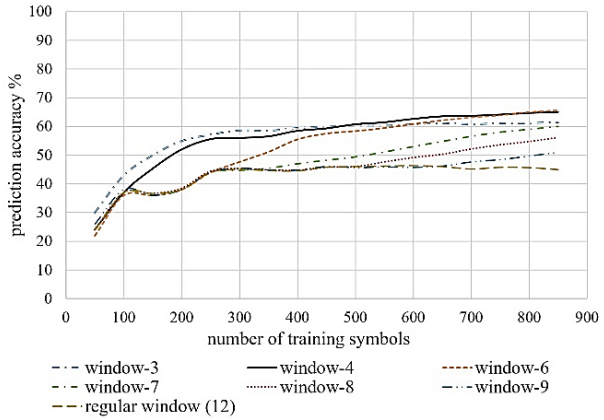


*Figure 4:* Comparison of Active LeZi algorithm with extended model of various window sizes (aggregated values after every 50 symbols)

### B. *Performance of Active LeZi with Constrained Window Size*

In this section, we compare the prediction accuracy of the Active LeZi algorithm with fixed (constrained) window length with the original Active LeZi algorithm. These experiments are performed on a synthesized training data of 850 symbols. Figure 4 shows the aggregate prediction accuracy percentage after every 50 symbols of data as well as a plot for '*regular window (12)*', which is the regular window size when there are no restrictions on the length of the window for the Active LeZi algorithm. We observed that the Active LeZi algorithm with fixed maximum window size performs better than the Active LeZi algorithm.

This indicates that the model starts to lose the Markov property when the history grows large. In other words, small additional memory helps in prediction but if the memory gets too large then it reduces the prediction accuracy. Figures 5 and 6 show the results of the fixed window size for different number of symbols in the training set. Based on these result, we conjecture that when the length of window is restricted to a smaller value, which is close to the number of unique symbols in the training dataset, then the algorithm performs better.

### C. *Performance of Active LeZi with Interrupted Patterns*

Here, we compare the prediction accuracy of the Active LeZi algorithm extended by interrupted patterns with Active LeZi algorithm and LeZi Update algorithm. These experiments are performed on a synthesized training data of 850 symbols. We observed that the extended Active LeZi algorithm performs better than the other algorithms by considering interrupted patterns. Figure 7 shows the plots when the aggregate prediction accuracy percentage is plotted after every 50 symbols of data.

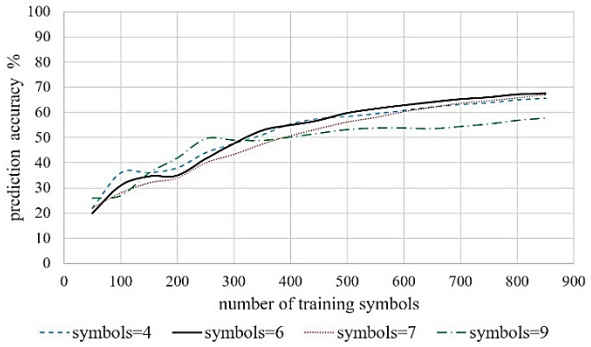


*Figure 5: Comparison of prediction accuracy of samples with varying number of unique symbols when the sliding window has a fixed size of 6 (aggregated values after every 50 symbols)*

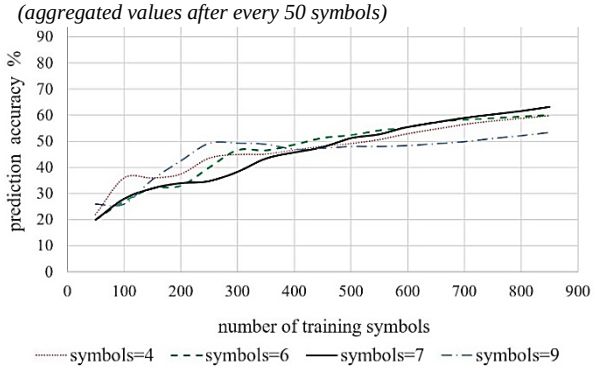


*Figure 6: Comparison of prediction accuracy of samples with varying number of unique symbols when the sliding window has a fixed size of 7 (aggregated values after every 50 symbols)*

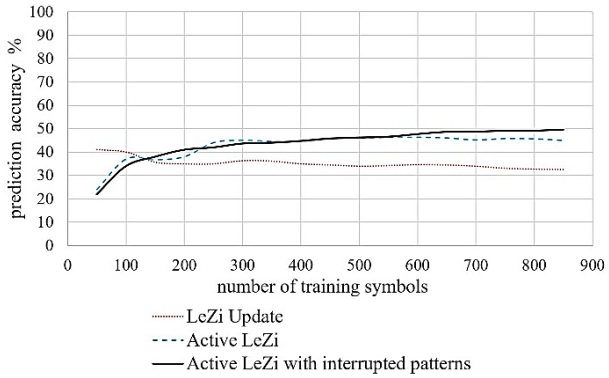


*Figure 7: Comparison of prediction algorithms (aggregated values after every 50 symbols)*

### *D. Performance of Time-Aware Active LeZi Algorithm*

In the earlier sections, we have seen that the Active LeZi algorithm performed better when the maximum size of the window is restricted to retain the Markov property. Thus, we now compare the performance of the Active LeZi algorithm with restricted window size against the Time-aware Active LeZi algorithm with restricted window size. Figure 8 shows the plots where the aggregate prediction accuracy percentage is plotted after every 20 symbols of data. We can notice the increase in prediction accuracy as the training sequence grows over time.

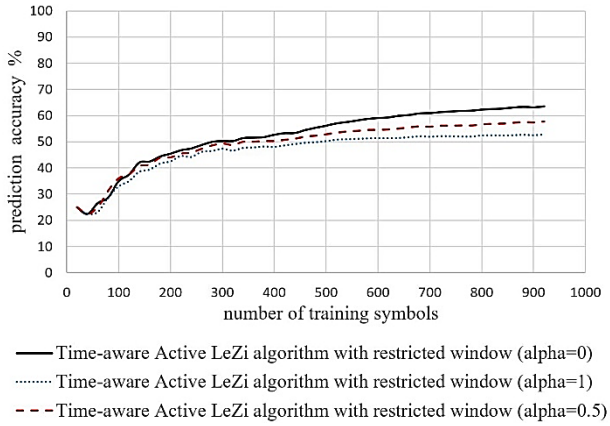


*Figure 8: Comparison of prediction accuracy of Time-aware Active LeZi with various α values (aggregated values after every 20 symbols)*

Time-aware Active LeZi calculates the probability of the next symbol based only on the occurrences in the respective time periods. Thus, even though it has a smaller training sequence in relative time slots, it performed better than the Active LeZi algorithm. The Time-aware Active LeZi algorithm generates the parse trie using the basis of the Active LeZi algorithm. It performs better when considering the frequency of the symbols at the appropriate time periods, as seen above. In an effort to experimentally evaluate the importance of the parameter 'α', we consider 3 cases: when 'α' has values 0, 0.5, and 1. In Figure 8, we see that the approach with $\alpha = 0$ performs better than the rest and the Time-aware Active LeZi algorithm performs better by itself. This is mainly due to the repeated patterns designed for our scenarios associated with each time interval. By repeating similar patterns between different time intervals, then the combination of the total frequency and time-aware frequency will lead to better prediction performance.

### *E. Performance Evaluation Based on Testbed Data*

In this section, we evaluate the performance of the Active LeZi algorithm, the Active LeZi algorithm with a fixed maximum length window, and Time-aware Active LeZi algorithm. Unlike the earlier tests, the data are collected from the testbed and contains 960 symbols representing a time span of 15 days, where each day has around 65 symbols and each time period has around 15 symbols. Figure 9 shows the plots where the aggregate prediction accuracy percentage is plotted after every 20 symbols of data. We notice that the time-aware Active LeZi algorithm performs better than the rest of the algorithms as the training sequence grows over time.

## VI. APPLICATIONS OF PREDICTIONS

Sequential prediction algorithms in the area of smart homes are suitable for applications such as energy efficiency and automation as well as security and health-care. We briefly introduce these application in this section.

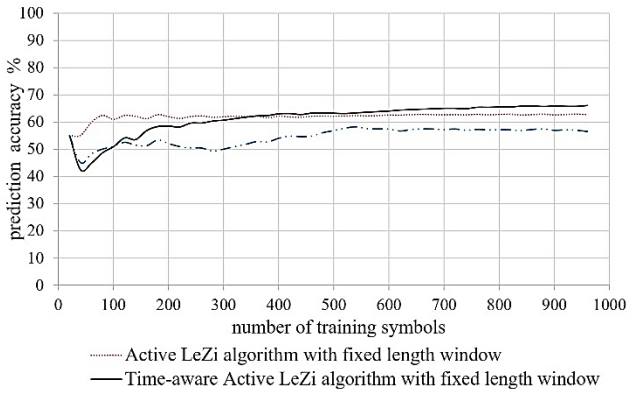


*Figure 9: Comparison of prediction accuracy of various prediction algorithms on testbed data (aggregated values after every 20 symbols)*

### *A. Energy System Automation*

The prediction model can be used to predict the future occupancy of the inhabitant, which can help in preheating/cooling the predicted areas of the smart home to maintain the comfort level temperature of the residents while avoiding wasting energy for the times that the inhabitants are not present. Such systems usually maintain the temperature at a certain set point to avoid delay in adjusting the temperature while saving energy. For example, in this paper, we consider the data set in Fig. 10 as the occupancy observed and the functioning of zone-wise systems [33] to control the conditions as in Figure 11.

```
abaccbaacbcabbaabbac
MMMMMAAAAAEEEEENNNNN
```

*Figure 10: Sample data with respect to time periods*

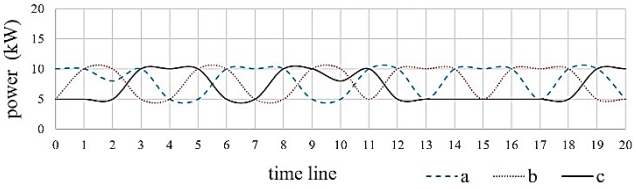


*Figure 11: The working of zone-wise systems to control temperature is explained as rise and drop in power levels according to the observations and predictions*

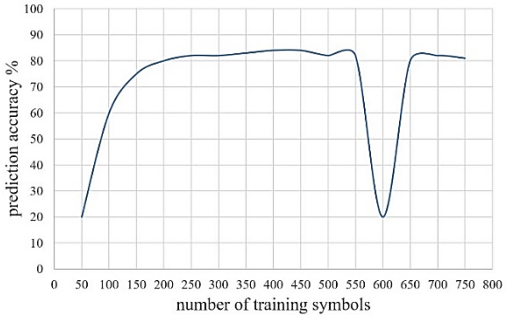


*Figure 12: Demonstration of intruder detection using prediction accuracy*

In this example, when the system predicts the symbol *b(A)* after *c(M)*, it starts to preheat the room b so that the room is adjusted to a comfortable temperature before the inhabitant occupies the room. In ideal cases, when all the predictions are accurate, this application can be extended to automation of lighting systems and appliances like coffee machine, entertainment systems, etc.

### B. *Security Applications*

Prediction models can also be used in security applications. For instance, large number of misses for occurrence of events based on the prediction algorithm trained using regular residents activities can trigger a security alarm as the pattern of the residents may differ with an intruder or an abnormal condition.

As observed in the earlier section, the prediction model attains optimum predictability as the training sequence grows larger over time. Thus, any sudden massive drop in the prediction accuracy can indicate an anomaly, which can be a security threat (Fig. 12).

### C. *Health Monitoring*

Sequential prediction techniques can be employed in health monitoring and assistance in smart home environments. Specifically for inhabitants suffering from dementia, prediction models can be designed to learn daily activity patterns and assist them by reminding the next routine activity based on the predictions. Another application of prediction can be assisting the elderly in medication. The device locations and usage patterns can be recorded simultaneously with the occupancy patterns. In this application, the prediction model is trained on the location and usage of the medical dispenser. The Time-aware Active LeZi algorithm can play a critical role in such applications by observing the occupancy of the room in which the dispenser is located as well as the usage of the dispenser throughout the day.

## VII. CONCLUSION AND FUTURE WORK

The existence and rise of smart homes has increased the need of applying intelligence in learning the activities of inhabitants and predicting their future demands in applications like resource management, energy efficient automation, security, and health monitoring. In this paper, we studied a sequential prediction model based on the Active LeZi algorithm. Specifically, we have evaluated the effects of the order of the memory in Markov chain model on the prediction accuracy. We showed that limiting the memory of the Markov chain model in the Active LeZi algorithm can improve the performance of the prediction. We also suggested extensions to include the effects of the interrupted patterns and the temporal information of the patterns in the prediction model. We evaluated the performance of the proposed approaches using a small-scale smart home testbed as well as synthesized and showed that the suggested changes can lead to better prediction performance for the Active LeZi algorithm.

## REFERENCES


[1] K. Gopalratnam and D. J. Cook, "Active LeZi: An Incremental Parsing Algorithm for Sequential Prediction," *IEEE Intelligent Systems*, pp. 52-58, 2007.

[2] A. Lotfi, C. Langensiepen, S. M. Mahmoud and M. J. Akhlaghinia, "Smart Homes for the Elderly Dementia Sufferers: Identification and Prediction of Abnormal Behaviour," *Journal of Ambient Intelligence and Humanized Computing*, pp. 205-218, 2012.

[3] J. C. Augusto and C. D. Nugent, "Smart Homes Can Be Smarter," in *Designing Smart Homes*, pp. 1-15, 2006.

[4] A. Fleury, M. Vacher and N. Noury, "SVM-Based Multimodal Classification of Activities of Daily Living in Health Smart Homes: Sensors, Algorithms, and First Experimental Results," *IEEE Transactions on Information Technology in Biomedicine*, pp. 274 - 283, 2010.

[5] S. P. Rao and D. J. Cook, "Predicting Inhabitant Action Using Action And Task Models With Application To Smart Homes," *International Journal on Artificial Intelligence Tools*, pp. 81-100, 2004.

[6] D. N. Monekosso, N. Dorothy and P. Remagnino, "Anomalous Behavior Detection: Supporting Independent Living," in *Intelligent Environments*, pp. 33-48, 2008.

[7] G. Singla and D. J. Cook, "Recognizing independent and joint activities among multiple residents in smart environments," *Journal of Ambient Intelligence and Humanized Computing*, pp. 57-63, 2010.

[8] N. Abe and M. K. Warmuth, "On the computational complexity of approximating probability distributions by probabilistic automata.," *Machine Learning*, pp. 205-260, 1992.

[9] J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, pp. 396-402, 1984.

[10] "www.wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/LZ77_and_LZ78. [Accessed: 20 - June - 2016]

[11] A. Bhattacharya and S. K. Das, "LeZi-update: an information-theoretic framework for personal mobility tracking in PCS networks," *Journal on Wireless Networks* , pp. 121-135, 2002.

[12] M. R. Alam, M. B. I. Reaz and M. A. M. Ali, "SPEED: An Inhabitant Activity Prediction Algorithm for Smart Homes," *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 985-990, 2012.

[13] M. Chetty, D. Tran and R. E. Grinter, "Getting to green: understanding resource consumption in the home," in *UbiComp '08 Proceedings of the 10th international conference on Ubiquitous computing*, 2008.

[14] J. Scott, A. B. Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges and N. Villar, "PreHeat: Controlling Home Heating Using Occupancy Prediction," in *UbiComp '11 Proceedings of the 13th international conference on Ubiquitous computing*, 2011.

[15] Y. Wang and P. Dasgupta, "Designing adaptive lighting control algorithms for smart buildings and homes," in *IEEE 11th International Conference on Networking, Sensing and Control*, Miami, 2014.

[16] L. Bonanni, E. Arroyo, C. Lee and T. Selker, "Smart sinks: real-world opportunities for context-aware interaction," in *Proceeding CHI '05 Extended Abstracts on Human Factors in Computing Systems*, 2005.

[17] A. R. Al-Ali, "Smart Home Renewable Energy Management System," in *The Proceedings of International Conference on Smart Grid and Clean Energy Technologies*, 2011.

[18] Y. M. Wi, J. U. Lee and S. K. Joo, "Electric vehicle charging method for smart homes/buildings with a photovoltaic system," *IEEE Transactions on Consumer Electronics*, pp. 323 - 328, 2013.

[19] M. Aditya, I. David, S. Navin, S. Prashant and T. Don, "The case for efficient renewable energy management in smart homes," in *BuildSys'11*, 2011.

[20] R. J. Robles and T.-h. Kim, "A Review on Security in Smart Home Development," *International Journal of Advanced Science and Technology*, pp. 13-22, 2010.

[21] A. R. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Pervasive Computing*, pp. 46-55, 2003.

[22] M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments," in *UbiComp 2002: Ubiquitous Computing*, 2002, pp. 237-245.

[23] S. A. Bagüés, A. Zeidler, F. Valdivielso and I. R. Matias, "Sentry@Home - Leveraging the Smart Home for Privacy in Pervasive Computing," *International Journal of Smart Home*, pp. 129-145, 2007.

[24] D. J. Freitas, T. B. Marcondes, L. H. V. Nakamura and R. I. Meneguette, "A Health Smart Home System to Report Incidents for Disabled People," in *International Conference on Distributed Computing in Sensor Systems*, Fortaleza , 2015.

[25] M. Skubic, G. Alexander, M. Popescu, M. Rantz and J. Keller, "A Smart Home Application to Eldercare: Current Status and Lessons Learned," *Journal on Technology and Health Care*, pp. 183-201, 2009.

[26] B. Ghazal and K. Al-Khatib, "Smart Home Automation System for Elderly, and Handicapped People using XBee," *International Journal of Smart Home*, pp. 203-210, 2015.

[27] M. Mozer, R. Dodier, D. Miller and M. Anderson, "Lessons from an Adaptive House," 2005. [Online]. Available: https://www.researchgate.net/profile/Michael_Mozer/publication/266594221_Lessons_From_An_Adaptive_House/links/55b90cdb08aed621de085fd9.pdf. [Accessed: 20 - June - 2016]

[28] D. J. Cook, M. Youngblood, I. Edwin O. Heierman, K. Gopalratnam, S. Rao, A. Litvin and F. Khawaja, "MavHome: An Agent-Based Smart Home," in *IEEE International Conference on Pervasive Computing and Communications* , 2013.

[29] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura and E. Jansen, "The Gator Tech Smart House: a programmable pervasive space," *Computer*, pp. 50 - 60 , 2005.

[30] G. D. Abowd, R. Orr, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner and W. A. Rogers, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research," in *Cooperative Buildings: Integrating Information, Organizations, and Architecture*, pp. 191-198, 1999.

[31] "Arduino Uno," [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed: 20 - June - 2016]

[32] "Hexbug Nano," [Online]. Available: https://www.hexbug.com/nano. [Accessed: 20 - June - 2016]

[33] J. G. Posa and B. H. Schwab, "Zone-based hvac system". Patent US20110253796 A1, 2011.