Use of Prediction Algorithms in Smart Homes

Aditi Dixit and Anjali Naik

***Abstract*—'Smart Homes' or 'Intelligent Homes' are capable in making smart or rational decisions and increase home automation. This is done to maximize inhabitant comfort and minimize operation cost. Tracing and predicting the mobility patterns and usages of devices by the inhabitant, sets a step towards the objective. The paper discusses in detail, the role of certain Prediction algorithms to bring about next event recognition. Further, an Episode Discovery helps in finding the frequency of occurrence of these events and targeting the particular events for automation. The effectiveness of the Prediction algorithms used is demonstrated ;making it clear how they prove to be a key component in the efficient implementation of a Smart Home architecture.**

***Index Terms*—Active LeZi, event history, LZ78, Smart Home, trie.**

I. Introduction

A 'Smart Home' is defined as a living or working space that interacts in a natural way and adapts to the occupant. Adaptation refers to the fact that it learns to recognize and change itself depending on the identity and activity undertaken by the occupant with minimal intervention from the occupant. Hence, a Smart Home agent must be able to predict the mobility patterns and device usages of the inhabitants. Using these predictions, the home can accurately route messages and multimedia information, and can automate activities that would otherwise be manually performed by the inhabitants. The Smart Home behaves as a rational agent, perceiving the state of the home through sensors and acting on the environment through effectors. The goal of the Smart Homes is to maximize comfort and safety, optimize energy usage and eliminate strenuous repetitive activities [1].

Prediction Algorithm helps in prediction of next probable state of the occupant and is a key component in developing an active Smart Home. Few Prediction Algorithms are IPAM,ONISI [2], Jacobs Blockeel Algorithm, LZ78 , LeZi-Update, Active LeZi, FXl and Adaptive FXl [3]. The choice of the Prediction Algorithm used will in turn affect the efficiency of working of a Smart Home. This paper talks about the use of Active LeZi Prediction Algorithm in logical implementation of a Smart Home.

Consider the following scenario, at 7:00 am, the alarm goes off ,which signals the bedroom night lamp to switch off and simultaneously the coffee maker in the kitchen to switch on. Ann heads to the bathroom, the news and weather forecast is displayed on the bathroom mirror. When Ann finishes grooming and heads towards the kitchen to have her morning coffee, the bathroom light goes off, the news program moves to the kitchen wall. When Ann leaves for work, the Smart Home secures the home and later that it places a grocery order for milk and bread. When Ann arrives from work, the grocery order has arrived .The Smart Home is prompt in recording minute details of interaction of Ann with her home every single minute.

Manuscript received October 9, 2013; revised December 13, 2013.

Aditi Dixit and Anjali Naikare are with Cummins College of Engineering For Women, Pune, India (e-mail: aditi.dxt@gmail.com).

II. Smart Home Architecture

The architecture of a Smart Home can be accurately depicted as four layers (Ref. Fig.1) :

Physical Layer: This layer contains the basic hardware within the house including individual devices, transducers, and network hardware.

Communication Layer: This layer includes software to format and route information between agents, between users and the house, and between the house and external resources.

Information Layer: This layer gathers, stores, and generates knowledge useful for decision making.

Decision Layer: This layer selects actions for the agent to execute based on information supplied from other layers.

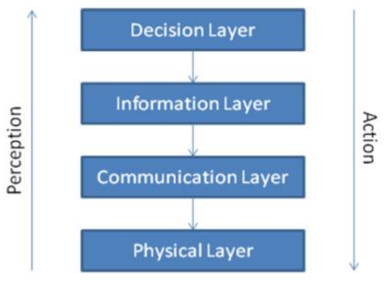


Fig. 1. Smart Home agent architecture.

Perception is a bottom-up process. The sensors monitor the environment (e.g. the temperature of the home) and, if necessary, transmit the information to another agent through the communication layer. The database records the information in the information layer, updates its learned concepts and predictions accordingly, and alerts the decision layer of the presence of new data.

The execution is a top down process. The decision layer selects an action (e.g. adjust the temperature to a lower value) and relates the decision to the information layer. After updating the database, the communication layer routes the action to physical layer. The physical allocates action to the appropriate effectors to execute [4].

III. Mobility Model

Seamless connectivity is a absolute necessity for a Smart Home. A wireless network system is incorporated in the entire home. The wireless terminals are usually integrated in the sensors deployed in a Smart Home environment and are to be worn by the inhabitant.

The Smart Home coverage area is partitioned into zones or sectors. Location management involves keeping track of the user movement. When Smart Home needs to contact an inhabitant, the system initiates a search for the target terminal device by polling all zones where it can possibly be found. All terminals listen to the broadcast page message, and only the target sends a response. The restrictions imposed by sensor characteristics like limitation due to infrared technology sometimes may become unavoidable. Also, to avoid location uncertainty of the inhabitant, a periodic update of the inhabitant's location needs to be captured. The most probable current position is predicted by update information and the last known position.

The Smart Home network can be represented by a bounded-degree connected graph, $G = (\vartheta, \varepsilon)$, where node set ϑ represents the zones and edge set ε represents the neighborhood (walls, hallways, etc.) between pairs of zones. Fig. 2. Shows the representation of a Smart Home by a graph.

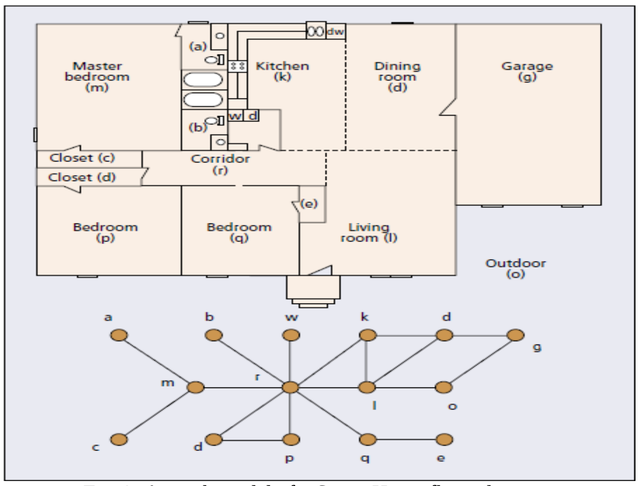
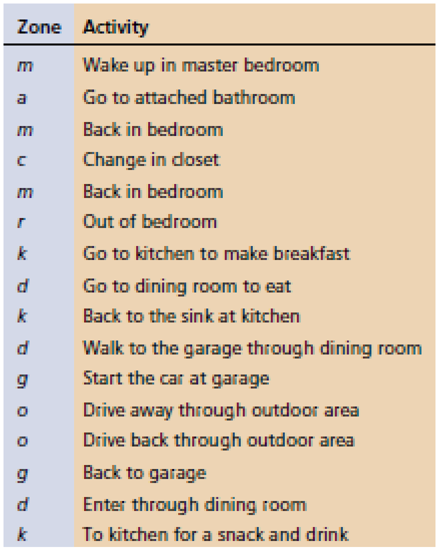


Fig. 2. A graph model of a Smart Home floor plan.



Zone	Activity
m	Wake up in master bedroom
a	Go to attached bathroom
m	Back in bedroom
c	Change in closet
m	Back in bedroom
r	Out of bedroom
k	Go to kitchen to make breakfast
d	Go to dining room to eat
k	Back to the sink at kitchen
d	Walk to the garage through dining room
g	Start the car at garage
o	Drive away through outdoor area
o	Drive back through outdoor area
g	Back to garage
d	Enter through dining room
k	To kitchen for a snack and drink

Fig. 3. Ann's Movement history.

The location determination of the inhabitant is a purely movement based scheme. An update is generated whenever a zone boundary crossing is detected or a distinctly different user activity as compared to the last one is observed.

The movement history of a user is represented by a string "$\upsilon_1\upsilon_2\upsilon_3$.." of symbols from the alphabet ϑ, where ϑ is the set of zones in the house and υ_i denotes the zone id reported by the i^{th} update. Consider, the movement history of Ann, in her Smart Home, during an entire day is generated as *mamcmrkdkd- googdk.*, where symbols denoting the movement history can be interpreted with the help of the Fig. 3. The tacit assumption is that an inhabitant's movement is merely a reflection of the patterns of his/her life, and those can be learned. This defines the learning phase that in turn aids decision making when reappearance of the patters are detected [4].

IV. Prediction Algorithms

The prediction algorithm aims to construct a universal predictor or estimator for determining the next user action. The scheme creates a dictionary of zone ids treated as character symbols and uses the dictionary to gather statistics based on movement history contexts, or phrases.

The problem of predicting the next symbol (representing a user action) in an input sequence can be formally defined as follows: Let $\sum$ be the set of possible input symbols and let $A = a_1 ... a_n$ with $a_j \in \sum$ be a sequence of input symbols of which the first i symbols, that is $a_1 \dots a_i$. A Prediction Algorithm decides at first whether it is able to make a prediction and if so, returns the probability for each symbol $x \in \sum$, that x is the next element in the input sequence. These values define a conditional probability distribution P over $\sum$, where $P(x|a_1 \dots a_i)$ is the probability for the singleton subset of $\sum$ containing x.

There are two ways of calculating the probabilities: on-demand or live. Algorithms using the former method maintain a data structure to compute the probabilities. They update the data structure after each symbol in the input sequence, whereas the live algorithms update the probability distributions itself [3].

A. LZ78 Algorithm

The LZ78 data compression is an incremental parsing algorithm based on the Markov model. This algorithm has been interpreted as a Universal modeling scheme that sequentially calculates empirical probabilities in each context of the data; the generated probabilities reflect contexts seen from the beginning of the parsed sequence to the current symbol. The LZ code length of any individual sequence attains the Markovian empirical entropy for any finite Markov order. This algorithm parses an input string "$x_{1,} x_2 \dots x_i$" into $c(i)$ substrings "$w_1, w_2, \dots w_{c(i)}$" such that for all j>0, the prefix of the substring w_j (i.e., all but the last character of w_j) is equal to some w_i for $1<i<j$. Because of this prefix property, parsed substrings can efficiently be maintained in a trie. The LZ78 is used only as a system that breaks up a given sequence (string) of states into phrases.

Algorithm:

```
initialize dictionary := null
  initialize phrase $w$ := null
  loop
     wait for next symbol $v$
    if (($w.v$) in dictionary):
            w := w.v
        else
            add (w.v) to dictionary
            w := null
            increment frequency for every
            possible prefix of phrase
        endif
    forever
```

Consider the sequence of input symbols *nx* = "*aaababbbbbaabccddcbaaaa*". An LZ78 parsing of this string of input symbols as per above mentioned algorithm of LZ78 ,would yield the following set of phrases: "*a,aa,b,ab,bb,bba,abc,c,d,dc,ba,aaa*". The generated dictionary is shown below. As described above, this algorithm maintains statistics for all contexts seen within the phrases w_i thesetext statistics are stored in a trie(Ref. Fig. 4.)

Dictionary:

a	5
aa	2
b	4
ab	2
bb	2
bba	1
abc	1
c	1
d	2
dc	1
ba	1
aaa	1

The LZ78 algorithm suffers from the slow convergence problem. This is because, all the information crossing phrase boundaries is lost. In many situations, there will be significant patterns crossing phrase boundaries, and these patterns will affect the next symbol in the sequence [5].

Eg. In above example string (*aaababbbbbaabccddcbaaaa*), 6th symbol (*bba*) and 7th (*abc*) does not form phrase *baab.*

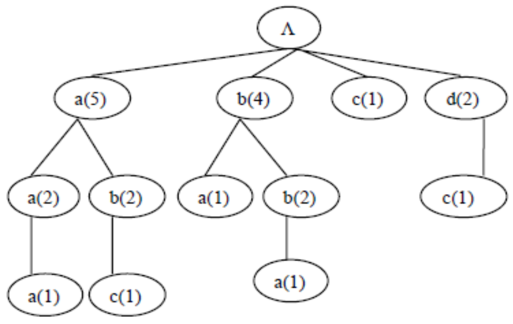


Fig. 4. Trie formed by LZ78 parsing of the string "*aaababbbbbaabccddcbaaaa*" (Order-2 Markov model).

B. *Active LeZi Algorithm*

The Active LeZi is an on-demand algorithm that is based on Markov models and primarily stores the frequency of input patterns in a trie according to the compression algorithm LZ78. The amount of information being lost across the phrase boundaries increases rapidly when there is an increase in the number of states seen in the input sequence. This problem can be overcome by maintaining a variable length window of previously-seen symbols.

The length of the longest phrase seen in a classical LZ78 parsing is chosen as equal to the length of window at each stage. The reason for selecting this window size is that the LZ78 algorithm is essentially constructing an order-*k-1* Markov model, where *k* is equal to the length of the longest LZ78 phrase seen so far.

Within this window, we can now gather statistics on all possible contexts. This builds a better approximation to the order-k Markov model, because it has captured information about contexts in the input sequence that cross phrase boundaries in the classical LZ78 parsing. Therefore, we gain a better convergence rate to optimal predictability as well as greater predictive accuracy [5].

Characteristics of Active LeZi are as follows:

- A growing-order Markov model attains optimal FS predictability, due to the optimality of LZ78.
- As the length of the longest LZ78 phrase grows, Active LeZi stores more and more information; as the input sequence (the experience) grows, the algorithm performs better. This is a desirable characteristic of any learning algorithm.

The given trie in Fig. 5 shows the Active LeZi parsing of the same string "*aaababbbbbaabccddcbaaaa*" as per the Active LeZi algorithm.

Algorithm:

```
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
```

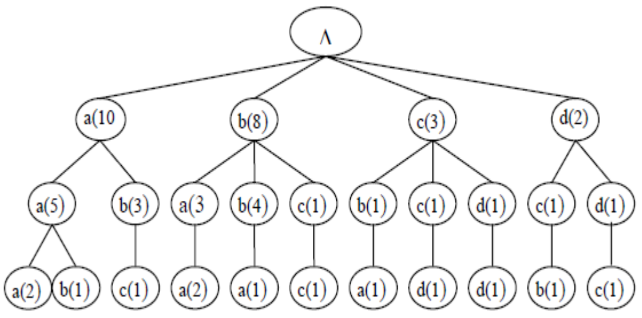


Fig. 5. Trie formed by Active LeZi parsing of the string "*aaababbbbbaabccddcbaaaa*"

The Active LeZi builds an order-k Markov model. The Prediction by Partial Match (PPM) strategy of *exclusion* to gather information from models of order 1 through k to assign the next symbol its probability value. Consider the example of previous sequence "*aaababbbbbaabccddcbaaaa*".

The window maintained by Active LeZi represents the set of contexts used to compute the probability of the next symbol. In the example, the last phrase "*aaa*" (which is also the current ALZ window) is used. Within this phrase, the contexts that can be used are all suffixes within the phrase, except the window itself (i.e. "*aa*," "*a*," and the null context).

Suppose the probability that the next symbol is an "*a"* is being computed. It is seen that(Ref. Fig. 5), an "*a"* occurs two out of the five times that the context "*aa*" appears, the other cases producing two null outcomes and one "*b*". Therefore the probability of encountering an "*a*" at the context "*aa*" is 2/5, and we now fall back to the order-1 with probability 2/5. At the order-1 context, we see an "*a*" five out of the ten times that we see the "*a*" context, and of the remaining cases, we see two null outcomes. Therefore, we predict the "*a*" at the order-1 context with probability 5/10, and escape to the order-0 model with probability 2/10. At the order-0 model, we see the "*a*" ten out of 23 symbols seen so far, and we therefore predict "*a*" with probability 10/23 at the null context.

The blended probability of seeing an "*a*" as the next symbol is therefore:

$$2/5 + 2/5 \times (5/10 + 2/10 \times (10/23))$$

The probability that the next symbol is a "*c*". In this case, the order-2 and the order-1 contexts do not yield a "*c*". Therefore, we escape to the order-0 model and predict a "*c*" with a probability of 3/23. In this case the total probability of seeing a "c" would be

$$0/5 + 2/5 \times (0/10 + 2/10 \times (3/23)) = 2/5 \times 2/10 \times 3/23$$

This method of assigning probabilities has the following advantages:

- It solves the zero-frequency problem. In the above example, if only the longest context had been chosen to make a decision on probability, it would have returned a zero probability for the symbol "*c*," whereas lower-order models show that this probability is indeed non-zero.
- This blending strategy assigns greater weight to higher-order models in calculating probability if the symbol being considered is found in that context, while lower-order models are suppressed owing to the null context escape probability [5].

V. Episode Discovery

The prediction results obtained from the prediction algorithm above can be used to automate interactions with the home, removing the need for manual control of devices. However, these automated actions can be annoying or detrimental if the inhabitant must undo the action executed by the house or repair damage caused by a faulty decision. To eliminate the possibility of frequent occurrence of this event, the episode discovery becomes a necessary part of Smart Home prediction logic [4], [6].

Instead of identifying and automating each inhabitant pattern, we use an Episode Discovery (ED) prediction algorithm that identifies significant episodes within an inhabitant event history. A significant episode can be viewed as a related set of device events that may be ordered, partially ordered, or unordered. A significant episode occurs at some regular interval or in response to other significant episodes called *triggers*. The objective of the problem domain is to mine the input stream in order to discover the significant episodes. Actions can then be automated based on the significance of the discovered pattern as well as the predictive accuracy of the next event [4].

The episode discovery problem is defined as follows. Let E be the set of all device events. An event occurrence O is a pair (e, t) relating an event e to an integer time value t. An event sequence S is an ordered sequence of event occurrences. The episode ε is defined as a set of event occurrences, and a candidate item set I as a set of events and episodes, $I = (\{e1, e2, e3, en\},\{E1, E2, \ldots, Em\})$, where each event ei has an occurrence in each of the episodes Ej, for $1 \le i \le n$ and $1 \le j \le m$. A significant episode L is an episode that meets or exceeds the evaluation threshold, and an event sequence description D is a description of an event sequence using significant episodes and event occurrences.

Algorithm for Episode Discovery:

1) Construct an event sequence S from input O.
2) Partition S into episodes using a sliding window of length w.
3) Create candidate itemsets from the episodes.
4) Compute compression values for each of the candidate itemsets.
5) Using a greedy approach, identify the candidate item set that minimizes the description length of the set of episodes as a significant episode.
6) Remove all of the episodes associated with the candidate itemset from the remaining candidate itemsets.
7) Remove all candidate itemsets that have an empty episode set.
8) Repeat steps 4–7 until the list of candidate [4].

The above described Episode Discovery algorithm helps to gain significant insights to the episodes and the frequency of occurrences.

For example, certain conclusions from these algorithms can be as-

- Alarm On, Alarm Off, Bedroom Light On, Coffeemaker On, Bathroom Light On (daily)
- Bedroom Light Off, Bathroom Light Off. Kitchen Light On, Kitchen Screen On (daily) Coffee Maker Off, Kitchen Light Off, Kitchen Screen Off (daily)
- Order Groceries (weekly)
- Other activities, such as switching on of Television, would not be identified by ED as significant because they do not occur with any predictable regularity [4].

VI. Analysis of Prediction Algorithm

The evaluations of performance of Prediction Algorithms, the following metrics are used in the literature:

- Prediction accuracy pr_{ac}
- Prediction probability pr_p
- Applicability a_p

The prediction accuracy and probability are computed by assigning a score to every prediction made by the algorithm and averaging over the number of predictions made by the algorithm. Thus, we can compute the prediction accuracy pr_{ac} and probability pr_p over the whole input sequence $a_1 \ldots a_n$ using following equation:

$$pr_x(a_1 \ldots a_n) = (1/m) \times \sum_{i=0}^{n-1} . \, eval_x(a_1 \ldots a_i, a_{i+1})$$

where m ($\leq n$) is the number of predictions made and $eval_x$ the evaluation function that returns the value for a single prediction (it returns 0 if no prediction was made by the algorithm). The $eval_x$ function thereby differs for the two metrics.

The prediction accuracy $eval_{ac}$ considers only the symbols which are predicted with maximal probability. We define the set A as the set of all these symbols:

$$A_{i+1} = \{x \in \_ | \forall y \in \_ . P(x|a_1 \ldots a_i) \geq P(y|a_1 \ldots a_i)\}.$$

The $eval_{ac}$ function for the prediction accuracy then returns a value reflecting whether the actual next symbol a_{i+1} is among these values A_{i+1}. Thus, $eval_{ac}$ is 0 if $a_i + 1 \in / A_{i+1}$ otherwise it is computed as $eval_{ac}(a_1 \ldots a_i, a_{i+1}) = 1/|A_{i+1}|$.

In contrast, the $eval_p$ function for the prediction probability rates with which probability the correct symbol was suggested, no matter if it has been assigned a maximal probability or not: $eval_p(a_1 \ldots a_i, a_{i+1}) = P(a_{i+1|} a_1 \ldots a_i)$.

The third metric applicability a_p is defined as the ratio of input symbols the algorithm was able to make a prediction for: $a_p = m/n$ [3].

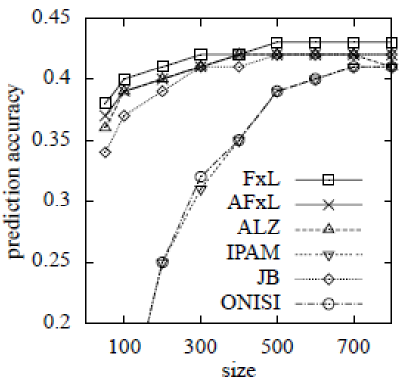


Fig. 6. Performance based on dataset size.

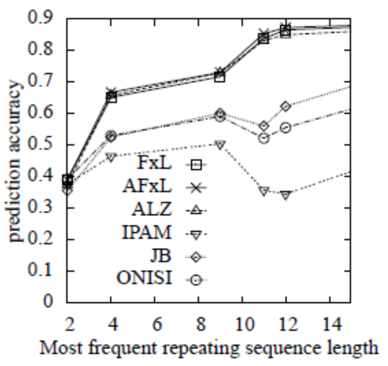


Fig. 7. Performance regarding sequence length.

The characteristics of the input sequences plays an important role to find which algorithm is best suited to requirements of the application. The most important parameters are:

- Dataset size available for training
- Distribution of repetitive sequences
- Noise in the repetitive sequences

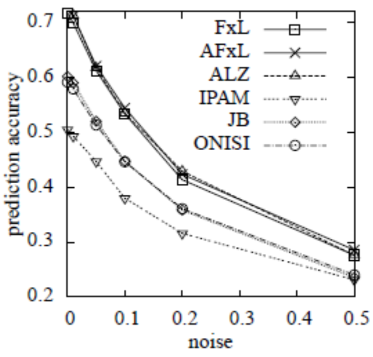


Fig. 8. Performance regarding different noise levels.

Fig. 6, Fig. 7 and Fig. 8 denote the results of an experiment carried out by Melanie Hartmann and Daniel Schreiber prove the significance of above mentioned characteristics of input sequences in determining prediction accuracy of various Prediction Algorithms.

- The Fig. 6 shows, for Active LeZi Prediction Algorithm, with an increase in the size of the dataset, there is an increase in the prediction accuracy.
- The Fig. 7 shows, for the Active LeZi Prediction Algorithm, there is an increase in the prediction accuracy with increase in the frequency of the repeating sequence.

The Fig. 8 shows, there is an overall decay in prediction accuracy with increase in noise in the input data. But we clearly see that, even in zones of higher noise, the Active LeZi comparatively demonstrates higher prediction accuracy than other algorithms like IPAM and ONISI.

Thus, from all the three graphs we conclude that Active LeZi displays an optimum performance regarding all tested parameters. The Active LeZi is a suitable choice for Smart Home logic implementation and is easy to understand and incorporate.

VII. Conclusion

In the paper, the concept of a Smart Home and necessary steps in implementing its logic is discussed. The skeletal architecture for implementation of a Smart Home provides an approach to its deployment. The paper sheds a light on three basic steps of logic development for a Smart Home. The first being creation of a mobility model and representation of event occurrences as strings. The second step, the use of a Prediction Algorithm to predict the most likely next state or inhabitant action. This step facilitates decision making for automation of necessary actions; is the most crucial step and backbone of the entire Smart Home framework. The LZ78 Algorithm is described, which primarily is used for a trie formation. The Active LeZi Prediction Algorithm is then introduced to overcome the drawbacks of LZ78 and helps in effective prediction of probable next event. The episode discovery helps to find significant patterns in event history and determine the frequency of its occurrence. It helps to identify which patterns can be automated easily with least fault occurrence. Lastly, we understand the parameters that influence the selection of a Prediction Algorithm. The Active LeZi Prediction Algorithm having suitable performance for all the tested parameters. It is easy to understand and implement and thus, an optimum choice for the logical implementation of a Smart Home.

REFERENCES

[1] D. N. Monekosso, P. Remagnino, and Y. Kuno, *Intelligent Environments: Methods, Algorithms and Applications*, Springer, 2009, ch. 1, pp. 1-11.

[2] P. Gorniak and D. Poole, "Predicting future user actions by observing unmodified applications," *AAAI-00 Proceedings*, 2000.

[3] M. Hartmann and D. Schreiber, "Prediction algorithms for user actions," *LWA'07*, pp. 349-354, 2007.

[4] S. K. Das, D. J. Cook, A. Bhattacharya, E. O. H. Iii, and T.-Y. Lin, "The role of prediction algorithms in the mavhome smart home architecture," *IEEE Wireless Communications*, December 2002.

[5] K. Gopalratnam and D. J. Cook, "Active Lezi: an incremental parsing algortihm for sequential prediction," *IEEE Intelligent Systems*, vol. 22, no. 1, pp. 52-58, 2007.

[6] S. Laxman, P. S. Shastry, and K. P. Unnikrishnan, "Fast algorithms for frequent episode discovery in event sequences," presented at the Third Workshop on Mining Temporal and Sequential Data Seattle, August 2004.

Aditi A. Dixit was born and brought up in Pune, India. She is a Ph.D. student of her final year in computer engineering at Cummins college of Engineering for women.

Her current research interests include artificial intelligence, neural networks, machine learning.

Anjali Naik is an assistant professor at Cummins College of Engineering for Women.

Her current research interest include artificial intelligence, neural networks, medical image processing.