MavHome: An Agent-Based Smart Home

Diane J. Cook, Michael Youngblood, Edwin O. Heierman, III,
Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja
Department of Computer Science Engineering
University of Texas at Arlington
Box 19015, Arlington, TX 76019-0015
Email: cook@cse.uta.edu

Abstract

The goal of the MavHome (**M**anaging **A**n Intelligent **V**ersatile **Home**) project is to create a home that acts as a rational agent. The agent's goal is to maximize inhabitant comfort and minimize operation cost. In this paper, we introduce the MavHome project and its underlying architecture. The role of prediction algorithms within the architecture is discussed, and a meta-predictor which combines the strengths of multiple approaches to inhabitant action prediction is presented. We demonstrate the effectiveness of these algorithms on sample smart home data.

1 Introduction

The MavHome smart home project is a multi-disciplinary research project at the University of Texas at Arlington focused on the creation of an intelligent and versatile home environment [2]. Our goal is to create a home that acts as a rational agent, perceiving the state of the home through sensors and acting upon the environment through effectors (in this case, device controllers). The agent acts in a way to maximize its goal, which is a function that maximizes comfort and productivity of its inhabitants and minimizes operation cost. In order to achieve these goals, the house must be able to predict, reason about, and adapt to its inhabitants.

MavHome operations can be characterized by the following scenario. At 6:45am, MavHome turns up the heat because it has learned that the home needs 15 minutes to warm to optimal temperature for waking. The alarm goes off at 7:00, which signals the bedroom light to go on as well as the coffee maker in the kitchen. Bob steps into the bathroom and turns on the light. MavHome records this interaction, displays the morning news on the bathroom video screen, and turns on the shower. While Bob is shaving MavHome senses that Bob is two pounds over his ideal weight and adjusts Bob's suggested menu. When Bob finishes grooming, the bathroom light turns off while the kitchen light and menu/schedule display turns on, and the news program moves to the kitchen screen. During breakfast, Bob notices that the floor is dirty and requests the janitor robot to clean the house. When Bob leaves for work, MavHome secures the home, and starts the lawn sprinklers despite knowing the 70% predicted chance of rain.

Later that morning, a rainstorm hits the area which further waters the lawn. Due to a nearby lightning strike, the VCR experiences a power surge and breaks down while taping Bob's favorite show. MavHome places a repair request and informs Bob at work of the event. Because the refrigerator is low on milk and cheese, MavHome places a grocery order to arrive just before Bob comes home. When Bob arrives home, his grocery order has arrived and the hot tub is waiting for him.

A number of capabilities are required for this scenario to occur. For a house to be able to record inhabitant interaction and trigger sequences of events such as the bedroom light / coffee maker sequence, advances in active database techniques are needed. Machine learning techniques are required to predict inhabitant movement patterns and typical activities, and to use that information in automating house decisions and optimizing inhabitant comfort, security, and productivity. In order for Bob's news program to follow him between rooms and for MavHome to find him away from the home, multimedia and mobile computing capabilities must be present. As can be observed from the scenario, MavHome automates the control of numerous devices within the home. To scale to this size problem, the MavHome agent can be decomposed into lower-level agents responsible for subtasks within the home, including robot and sensor agents, and MavHome can dynamically reorganize the hierarchy to maximize performance.

The desired smart home capabilities must be organized into an architecture that seamlessly connects these components while allowing improvement in any of the underlying technologies. In this paper we present such an architecture that supports the MavHome smart home project, and describe the role of the Predict^2 meta-predictor algorithm within this architecture. We validate the underlying algorithms on synthetic and real captured smart home data.

2 MavHome Architecture

The MavHome architecture is a hierarchy of rational agents which cooperate to meet the goals of the overall home. Figure 1 shows the architecture of a MavHome agent. The technologies within each agent are separated into four cooperating layers. The **Decision** layer selects actions for the agent to execute based on information supplied from the other layers through the Information layer. The **Information** layer gathers, stores, and generates knowledge useful for decision making. The **Communication** layer facilitates the communication of information, requests, and queries between agents. The **Physical** layer contains the hardware within the house including individual devices, transducers, and network hardware. Because the architecture is hierarchical, the Physical layer may actually represent another agent in the hierarchy.

Perception is a bottom-up process. Sensors monitor the environment (e.g., lawn moisture level) 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. During action execution, information flows top down. The Decision layer selects an action (e.g., run the sprinklers) and relates the decision to the Information layer. After updating the database, the Communication layer routes the action to the appropriate effector to execute. If the effector is actually another agent, the agent receives the command through its effector as perceived information and must decide upon the best method of executing the desired action. A specialized interface agent provides interaction capabilities with users and with external resources such as the Internet. Agents can communicate with other agents using the hierarchical flow of control and information shown in Figure 1.

Several smart home-related projects have been initiated by research labs. The Georgia Tech Aware Home [5] and the MIT Intelligent Room [9] include an impressive array of sensors to determine user locations and activities within an actual house. The Neural Network House at the University of Colorado Boulder [6] employs a neural network to control heating, lighting, ventilation, and water temperature in a manner that minimizes operating cost. The interest of industrial labs in smart home and networked appliance technologies is evidenced

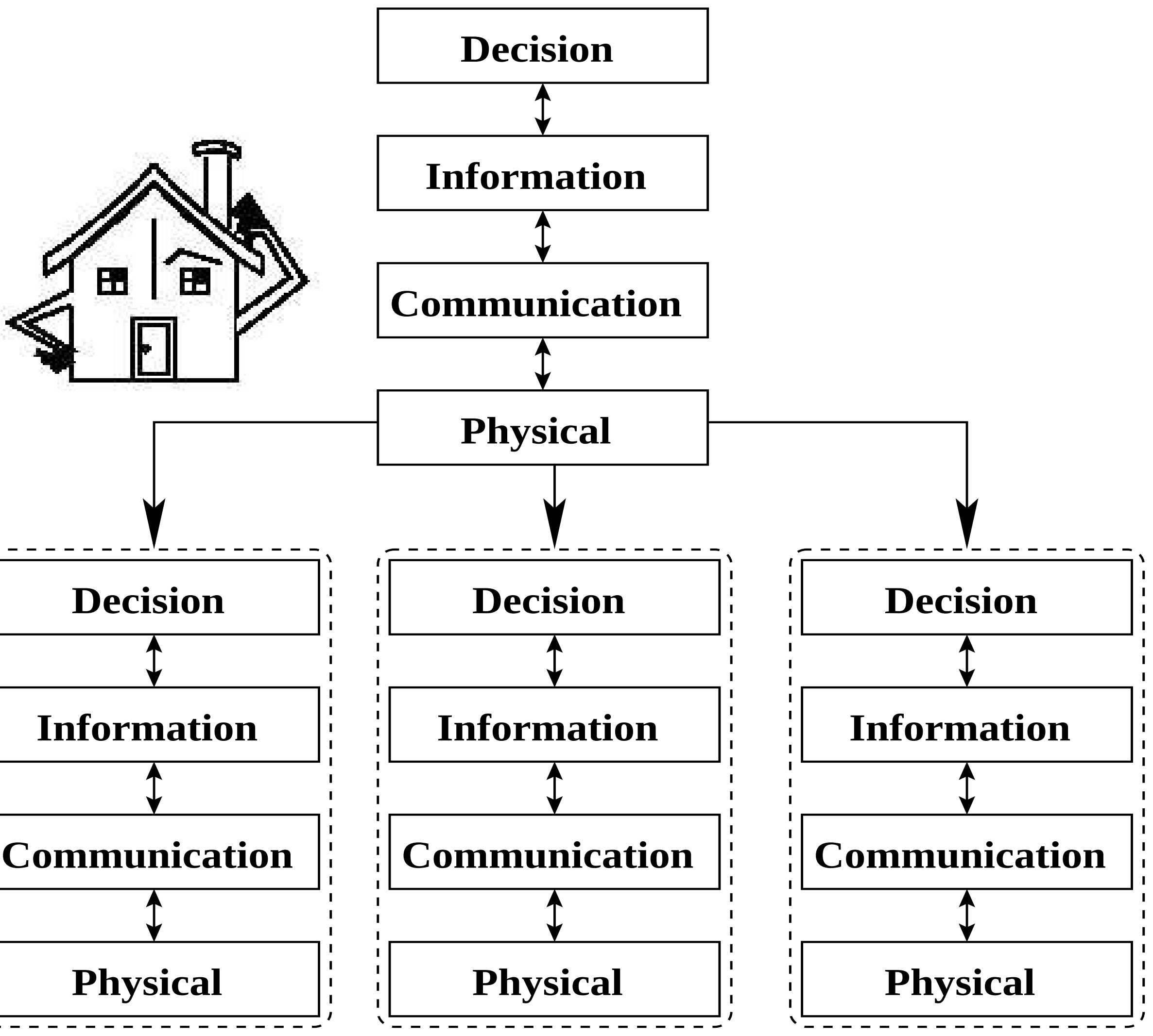

Figure 1: MavHome agent architecture.

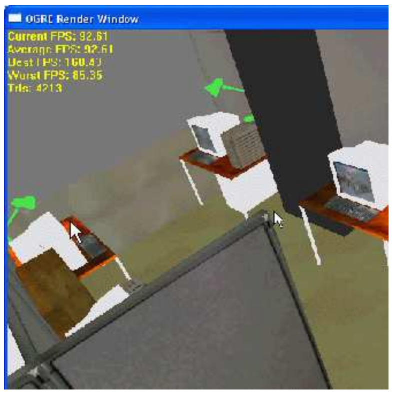

Figure 2: ResiSim simulator for the MavHome environment.

by the creation of Jini, Bluetooth, and SIP (Session Initiation Protocol) standards, and by supporting technologies such as Xerox PARC's Zombie Board, Microsoft's Home project, the Cisco Internet Home, and the Verizon Connected Family project. MavHome is unique in combining technologies from artificial intelligence, machine learning, databases, mobile computing, robotics, and multimedia computing to create an entire smart home that acts as a rational agent.

3 MavHome Implementation

The MavHome architecture has been implemented using a CORBA interface between software components and powerline control for most electric devices. In the current MavHome lab environment, students register their presence and MavHome begins collecting data on their activities. MavHome features include database collection of activities, prediction of inhabitant actions, identification of inhabitants from observed activities, mobility prediction, robotic assistants, multimedia adaptability, and intelligent control of the house. We also designing a 3D simulator of the environment, shown in Figure 2. Using the simulator, visitors at a remote location can monitor the status of the environment. Changing the status of the device in the simulator will change the status of the device in the physical environment as well.

One challenge is the automation of devices that are not controllable in their natural form. Inhabitants of MavHome want to be able to control the amount of natural light

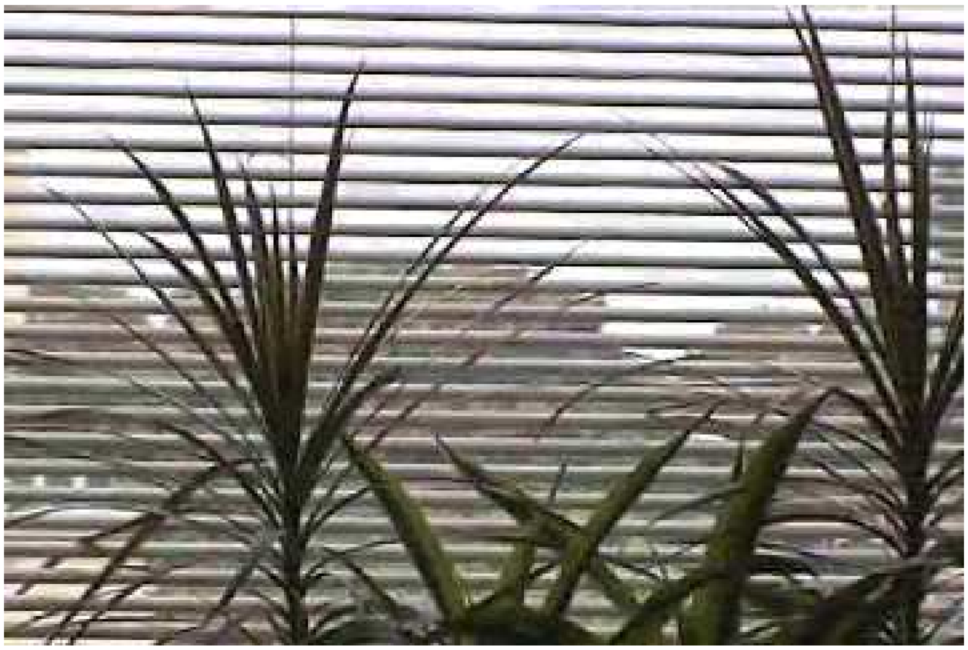

Figure 3: Automated blinds in MavHome environment.

through automation of mini-blinds. However, most blinds are not designed for such control. MavHome's controller hardware is designed to let the house turn single or multiple sets of mini-blinds clockwise or counter-clockwise a specific number of turns. The rod rotation hardware is built using stepper motors from 5 1/4" floppy drives, and a parallel port provides the interface between the MavHome computer and the hardware circuitry. A 4-16 decoder / de-multiplexer is used for selection and an inverter is used to convert from negative to positive logic. Opto-isolaters and transistors are incorporated to meet the power demands, and a Darlington driver is connected to four data bits of the parallel port to switch current to the coils in the stepper motor. CAT-5 cable transfers signals to the motor. The software uses threads to process multiple blinds independently. Because the software interface uses CORBA, these blinds can be automated by MavHome in the way any other device can be controlled by the intelligent agent. Figure 3 shows the blinds in the MavHome environment after being partially opened by the home agent. Similar approaches can be taken to automate desired features of the house with optimal modularity.

4 Inhabitant Action Prediction

Automating devices within a home is not sufficient to label it an intelligent environment. An intelligent environment must be able to acquire and apply knowledge about its inhabitants in order to adapt to the inhabitants and meet the goals of comfort and efficiency. These capabilities rely upon effective prediction algorithms. Given a prediction of inhabitant activities, MavHome can decide whether or not to automate the activity or even improve upon the standard activity sequence to meet the house goals.

Specifically, the MavHome intelligent agent needs to predict the inhabitant's next action in order to automate the routine and repetitive tasks for the inhabitant. Patterns observed in past inhabitant activities can be used to aid the agent decisions for controlling devices throughout the home. In this section we describe this role of prediction in the MavHome architecture and present a variety of prediction algorithms suitable for that task and controlled by the Predict^2 meta-predictor.

Prediction is a heavily researched area in artificial intelligence. The ONISI system [3] and the Korvemaker and Greiner UNIX command prediction algorithm [4] employ pattern matching. IDHYS [7] represents an approach to action prediction based on the Candidate Elimination algorithm.

4.1 Smart Home Inhabitant Prediction (SHIP)

Our SHIP algorithm matches the most recent sequence of events with sequences in collected histories. In the SHIP algorithm, the inhabitant commands are encapsulated using *actions* and *matches*. When the inhabitant issues a command to a device, it is recorded as an action in the inhabitant *history*. A match identifies a sequence in history that matches the immediate event history (a sequence ending with the most recent event). A match queue is maintained to ensure a match time that is close to linear.

The algorithm SHIP consists of two steps. First, the match queue is updated when a new action is recorded. At time t in state s we compute $l_t(s, a)$, the length of the longest sequences that end with action a in state s and match the history sequence immediately prior to time t. In addition, we define a frequency measure $f(s, a)$ which represents the number of times the action a has been taken from the current state. In the second step, the matches in the queue are evaluated based on the match length and frequency. SHIP ranks matches based on a combination of normalized match frequency and match length, according to the equation

$$R_t(s,a) \;=\; \alpha \frac{l_t(s,a)}{\Sigma_t l_t(s,a_i)} \;+\; (1-\alpha)\frac{f(s,a)}{\Sigma_t f(s,a_i)}.$$

SHIP returns the action a that is indicated by the match (s,a) with the greatest $R_t(s,a)$ value as its prediction.

To allow for gradual changes in the inhabitant patterns over time, the value of a matched pattern can be multiplied by a user-specified decay factor. The user also has the flexibility of weighting the match length and match frequency factors that affect a match value. Because an inhabitant pattern is likely to contain small variations between occurrences, an inexact match is employed to find sequence matches.

SHIP has been tested using synthetic smart home data and on real data collected by students using X10 controllers in their home. The synthetic data generator allows a variable number of devices and any number of scenarios, each of which contains actions that execute in a partially-ordered or totally-ordered sequence, with probabilistic ranges of execution times.

In these experiments, SHIP yields a predictive accuracy as high as 53.4% on the real data and 94.4% on the synthetic data. SHIP's performance on the real data climbs to over 80% if we consider the top three matches identified by SHIP.

SHIP is one of the prediction algorithms used by MavHome. The advantages of this algorithm are its design simplicity and ability to adapt to changing patterns. A key disadvantage of the algorithm is the fact that the entire action history must be stored and processed off line to return a prediction. This is not practical for large prediction tasks over a long period of time. In addition, the algorithm is currently designed to perform to predict the next event in a sequence and not the time at which the event will occur.

5 Prediction Using Active LeZi (ALZ)

Our second prediction algorithm, Active LeZi, uses information theory principles to process historical action sequences. Because we characterize inhabitant-device interaction as a Markov chain of events, we can utilize a sequential prediction scheme that has been shown to be optimal in terms of predictive accuracy for this type of prediction problem. Because ALZ is based on LZ78 data compression, which is an incremental parsing algorithm, it is also an online algorithm.

The LZ78 text compression algorithm incrementally parses an input string $s_1, s_2, \ldots, s_i$ into $c(i)$ substrings $w_1, w_2, w_{c(i)}$ such that for all $j > 0$, the prefix of the substring w_j is equal to some w_i for $1 < i < j$. The algorithm maintains statistics for all contexts of each phrase [10]. This information is used to compress and reconstruct text strings in an online fashion.

Active LeZi enhances the original LZ78 algorithm by recapturing information that would be lost across phrase boundaries. To do this, we maintain a variable-length window of processed symbols. The length of this window is equal to the length of the longest phrase seen so far. We gather statistics on all of the possible contexts seen based on this information. The resulting algorithm also gains a better rate of convergence to optimal predictive accuracy.

To perform prediction, ALZ calculates the probability of each action occurring in the parsed sequence, and predicts the action with the highest probability. To enhance prediction, ALZ incorporates ideas from the Prediction by Partial Match (PPM) family of predictors. This has been used to great effect by Bhattacharya, et al. [1] in a predictive framework based on LZ78. PPM algorithms consider different order Markov models to build a probability distribution by weighing the different order models appropriately. ALZ starts by building an order-k Markov model, then employs the PPM strategy to gather information from all lower-order models to determine the probability value of the next symbol.

Active LeZi has been tested on synthetic data generated for 30 days representing action sequences for weekday and weekend scenarios. The algorithm yielded 87% accuracy over this test data.

The ability of Active LeZi to use information from different order models is a strength of this approach, as is the ability to process information in an incremental manner. However, this algorithm does not incorporate actual event times into the prediction. For predictions that require this information the TMM algorithm is useful.

6 Prediction Using a Task-based Markov Model (TMM)

The TMM algorithm identifies high-level tasks in action sequences to help direct the creation of a Markov Model for action prediction. A simple Markov Model can be generated from collected action sequences and used to predict the next action given the current state of the agent. The state information captures the individual device states as well as the time of day and action date. However, additional information about the context in which activities were performed may be useful in further directing the prediction.

To this end, TMM identifies high-level tasks from the input action sequences. The sequence of events, or actions, is first partitioned into individual tasks. A change in tasks is identified by gaps in activities and changes in location of the actions being performed.

Second, a k-means clustering algorithm is used to cluster the partitioned task sequences into sets of similar tasks. Task meta-features are supplied to the clusterer that include the length in time and length in number of actions of the task sequence, the number of devices used in the task, and the number of locations covered by the task. The output of the clustering algorithm is a collection of task sets, each of which can be labeled as a separate task type. The task information can be used to refine the initial Markov Model by seeding probabilities of transitions between states based on their co-membership in the task cluster.

The TMM algorithm was tested on synthetic data generated for 30 days, using separate scenarios for weekdays and weekends. The algorithm generated 74% predictive accuracy on this data. Note that while the predictive accuracy is lower than for SHIP and Active LeZi, the complexity of the problem is increased because of the reasoning about time as well as action sequences.

7 Improving Prediction with Episode Discovery (ED)

The SHIP, Active LeZi, and TMM algorithms are useful in identifying likely activities of a smart home inhabitant. This information can be used to automate interactions with the home, removing the need for manual control of devices. A wrong prediction, however, can be annoying or detrimental if the inhabitant must reverse the action executed by the house or repair damage caused by a faulty decision.

Instead of identifying and automating each inhabitant pattern, we describe here a data mining algorithm, called Episode Discovery (ED), 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 goal of the intelligence framework in our 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.

Our approach is based on the work of Srikant and Agrawal [8] for mining sequential patterns from time-ordered transactions. Our home automation problem differs from previous research in that the input sequence does not consist of explicit transactions, but merely interactions with home devices. Unlike the previous sequence mining problem, the significant episodes in an intelligent environment may be ordered or unordered. In addition, many of the episodes in our environment will occur daily or weekly, and need to be recognized for this regularity. In our MavHome scenario, the following device activity sequences occur on a regular basis and should be detected by our algorithm:

- HeatOn (daily)
- AlarmOn, AlarmOff, BedroomLightOn, CoffeeMakerOn, BathRoomLightOn, BathRoomVideoOn, ShowerOn, HeatOff (daily)
- BedroomLightOff, BathRoomLightOff, BathRoomVideoOff, ShowerOff, KitchenLightOn, KitchenScreenOn (daily)
- CoffeeMakerOff, KitchenLightOff, KitchenScreenOff (daily)
- HotTubOn (daily)

- HotTubOff (daily)
- SprinklerOn (weekly)
- SprinklerOff (weekly)
- VCROn (weekly)
- VCROff (weekly)
- OrderGroceries (weekly)

Other activities, such as the robot activation, would not be identified by ED as significant because they do not occur with any predictable regularity.

To mine the data, the input sequence is partitioned into transaction-like collections of events by sliding a window over the event history and viewing the collection of events within the window as an ordered or unordered set. The minimum description length (MDL) principle is used to evaluate potential sequences. The MDL principle targets patterns that can be used to minimize the description length of the database by replacing each instance of the pattern with a pointer to the pattern definition. This evaluation measure thus identifies patterns that balance frequency with pattern length. As a result, automating these sequences will significantly reduce the amount of necessary interaction between an inhabitant and the environment. Another feature of ED is that patterns are evaluated for day, week, and month regularity as well as MDL value. Significant episodes will then be selected based on the overall evaluation measure, and used as the basis for activity prediction and home automation.

To improve the quality of predictions, action sequences are first filtered using ED. If a sequence is considered significant, then predictions can be made for events within the sequence window. We hypothesize that predictive accuracy will improve for any prediction algorithm that uses ED to filter the prediction set.

An online incremental version of ED has been implemented using C++. We tested our approach on a synthetic data set of 838 event occurrences generated from twenty-six unique possible events. Sixteen of the events are part of significant episodes, eight are noise, and two are both noise and part of a significant episode. ED identified nine of the significant episodes, which varied in length from one to five events. Four episodes were totally-ordered daily sequences, four were unordered daily occurrences, and one was a totally-ordered weekly scenario. We used ED as a filter for a simple sequence-based prediction algorithm based on frequency of (state, action) pairs.

The original prediction algorithm achieved 47% predictive accuracy. Enhancing the algorithm with ED, the predictive accuracy was 100%. These results show that ED can be used to aid in the automation of device interactions, as described by our MavHome scenario.

8 Predict^2

As is evident from our discussions, there are many approaches to prediction in the context of a smart home. The algorithms offer unique strengths and weaknesses, and no single algorithm appears to be best for all situations. Because we want to draw from all of the strengths, our main MavHome prediction algorithm is actually a meta-predictor, called Predict^2.

The Predict^2 algorithm uses a backpropagation neural network to learn a confidence value for each prediction algorithm based on the historical data gathered so far and accuracy of the algorithm on this data, along with meta features such as the amount of training data, the number of devices in the environment, the number of inhabitants in the home, and whether the current state is part of an ED-tagged significant episode. A voting scheme based on weighted votes from each individual prediction algorithm algorithm is used to generate the final prediction. A CORBA interface between Predict^2 and the MavHome architecture will allow the MavHome agent to draw upon the prediction when needed.

9 Conclusions

In this paper we have presented the MavHome smart home architecture, which allows a smart home (or other intelligent environment) to act as a rational agent. As a rational agent, the home receives input from sensors and selects an appropriate action which is executed through the use of effectors. This architecture allows the integration of research in machine learning, databases, mobile computing, robotics, and multimedia computing that is essential for smart home development.

As part of the MavHome architecture, several prediction algorithms are introduced that play critical roles in an adaptive and automated environment such as MavHome. The first prediction algorithm, SHIP, uses sequence matching with inexact allowances and decay factors to determine the most likely next inhabitant interaction with the home. The second algorithm, Active LeZi, enhances the LZ78 text compression algorithm to improve predictive accuracy and adapts the algorithm for probabilistic prediction of action sequences. The TMM algorithm identifies high-level tasks to aid in predicting actions with timing information. Finally, the ED algorithm mines the data using the principle of minimum descriptionlength to determine which episodes are significant enough to warrant prediction and automation. Results from synthetic and real collected smart home data indicate that the predictive accuracy is high even in the presence of many possible activities. The Predict^2 algorithm allows each of these prediction approaches to play a role in predicting inhabitant activity within MavHome.

We have demonstrated the effectiveness of these algorithms on collected data. The next step for this effort will be to implement the architecture in the context of actual smart environments. In these contexts we will show the effectiveness of the MavHome architecture operating as a rational agent, and its ability to improve the lifestyle of inhabitants in a variety of intelligent environments.

10 Acknowledgements

This work was supported by National Science Foundation grants IIS-0121297 and EIA-9820440.

References

[1] A. Bhattacharya and S. Das. Lezi-update: An information-theoretic framework for personal mobility tracking in PCS networks. *Wireless Networks Journal*, 8(2-3):121–135, 2002.

[2] S. Das, D. J. Cook, A. Bhattacharya, I. E O Heierman, and T.-Y. Lin. The role of prediction algorithms in the mavhome smart home architecture. *IEEE Personal Communications*, 2002.

[3] P. Gorniak and D. Poole. Predicting future user actions by observing unmodified applications. In *National Conference on Artificial Intelligence*, 2000.

[4] B. Korvemaker and R. Greiner. Predicting UNIX command lines: Adjusting to user patterns. In *Proceedings of the Conference on Intelligent Applications of Artificial Intelligence*, 2000.

[5] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, X. Ping, and S. X. Zhang. The intelligent home testbed. In *Proceedings of the Autonomy Control Software Workshop*, 1999.

[6] M. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proceedings of the AAAI Spring Symposium on Intelligent Environments*, pages 110–114, 1998.

[7] J.-D. Ruvini and C. Dony. Ape : Learning users habits to automate repetitive tasks. In *International ACM Conference on Intelligent User Interfaces*, pages 229–232, 2000.

[8] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the Fifth International Conference on Extending Database Technology (EDBT)*, 1996.

[9] M. C. Torrance. Advances in human-computer interaction: The intelligent room. In *Working Notes of the CHI 95 Research Symposium*, 1995.

[10] J. Ziv and A. Lempel. Compression of individual sequences via variable rate coding. *IEEE Transactions on Information Theory*, IT-24:530–536, 1978.