Improvements in data storage and tree generation in modified-SPEED Algorithm

Mamun Bin Ibne Reaz [a,1,*], Araf Farayez [a]

[a] Centre of Advanced Electronic and Communication Engineering, Faculty of Engineering and Built Environment,Universiti Kebangsaan Malaysia, Bangi, Malaysia
[1] mamun@ukm.edu.my *
* Corresponding Author

Received 28 April 2021; Accepted 4 May 2021; Published 31 May 2021

ABSTRACT

With the rising demand for smart devices and smart home systems, automation and activity prediction have become a vital aspect of people's everyday lives. Researchers have focused on developing approaches which detect patterns in user activities and used them to predict future actions. One such system is Modified Sequence Prediction via Enhanced Episode Discovery (M-SPEED), that uses spatiotemporal data of activities of daily lives to analyze user behaviors. But the low accuracy of this algorithm can be a limiting factor in efficient activity prediction. Also, the computational overhead of run time and memory causes this algorithm to show poor performance in case of large datasets. This research focuses on modifying the M-SPEED algorithm to improve its capability to run on larger dataset while at the same time improving run time. The accuracy is also improved to make it more effective in real-world applications. Proof of algorithm efficiency is provided to ensure system validity, and simulation is carried out on real-life data. The results demonstrate a 66.69% improvement in cumulative memory efficiency, 37% faster run time, and 8.22% better accuracy confirming the effectiveness of the proposal.

KEYWORDS
Tree generation
SPEED
M-SPEED
Episode discovery

1. Introduction

Smart home technology has become the 21st century life style, that integrates tech and its users. For the 15% handicapped population and a growing number of elderly people all over the world, smart home has already become a necessity [1]. It helps them to stay at home for longer, making their regular lives more comfortable and self-reliant. Beginning its journey only a decade ago, smart home technology has evolved into a multidimensional computing process over the last few years. It aims to benefit not only the handicapped and elderly population of the world, but also different people coming from different walks of life. With each passing year, the interactive technology is becoming more pervasive and accurate in its predictions.

Over the past few decades numerous researches are conducted to develop an effective activity prediction system to deliver automation in smart home environments [2]. In this quest, data compression algorithms gained enormous recognition among researchers due to their pattern predicting capabilities [3]. One of such is the LZ78. Bhattacharya and Das used the LZ78 dictionary contexts to develop the LeZi update algorithm [4]. This approach assists in mobility tracking in smart spaces by predicting user movements. But due to data loss across phase boundaries and slow convergence rate, the LZ78 renders a significantly poor performance in real world applications [5]. The data loss is later improved by Gopalaratnam and Cook through introduction of a variable length window in the improved algorithm Active LeZi (ALZ) [6]. This approach associates more weight to recent activities thus adding higher sensitivity towards latest data. Vikramaditya and Cook further developed temporal rules for the ALZ which improved the prediction accuracy and reduced error rate by 1% for real data [7].

Alam et al. developed the novel SPEED (Sequence Prediction via Enhance Episode Discovery) algorithm utilizing the Prediction by Partial Matching (PPM) data compression method [8]. This algorithm uses episodes, comprised of sequential events and behavioral patterns which are arranged in a finite order Markov model. By achieving an accuracy of 88.3%, it performs better than LeZi Update and ALZ algorithm. The SPEED is further improved by Marufuzzaman et al. by incorporating the time component [9]. The resultant M-SPEED (Modified-SPEED) can eliminate false episode detection and demonstrate a higher prediction accuracy. Further enhancements are added to this process by introduction of the location component in order to specifically carry out particular actions based on individual's location in the house [10]. A more recent algorithm SPADE (Sequence Prediction via All Discoverable Episodes) modifies the M-SPEED in tree generation phase to further improve the accuracy and runtime [11].

PPM has a higher computational complexity compared to LZ78 and other data compression approaches [12]. In the implementation of the SPEED algorithm, this complexity impacts the context generation and linear storage of generated contexts. Due to exponential memory growth, this algorithm is unable to process recent datasets like CASAS and ARAS [13], [14]. The primary goal of this research is to address this issue and improve the performance of the SPEED algorithm, both in terms of memory and run time.

2. Method

Smart home provides automation and comfort through the collection of user activities, using sensors distributed throughout the house. An even distribution of sensors leads to a better prediction of user actions. A wide variety of sensors are preferred, such as, motion sensors, power sensors, door sensors, cabinet sensors, etc. Data from these sensors are first stored in current databases and later passed on to the prediction engine in bulk. This data contains activities of daily life (ADLs).

In the first step, appropriate sequences are created from analyzing raw data. Here, the state info is merged with the sensor codes by exploiting opposite cases of English alphabet. For ON state, the sensor code is transformed into uppercase letter, and for OFF state, lowercase. Room number is combined with the sensor codes and a limit is set for the maximum number of sensors. The final sequence attains the following format:

[Sensor Event][Room number]@[Time]

In the next step, episodes are generated from these sequences. Episode generation process developed by Alam et al. is implemented without any further modification [8]. Generated results are stored in array along with their respective time intervals. This array is passed to the next module for extraction of all-possible contexts.

Traditionally, the SPEED algorithm uses a linear 2-dimensional array to hold the generated possibilities [9]. For an episode 'Abca', the possibilities are 'A', 'b', 'c', 'a', 'Ab', 'bc', 'ca', 'Abc', 'bca', and 'Abca'. It is evident from this example that multiple repetitions may occur in the possibility array depending on data diversity. A prefix tree based approach is suggested in this research to eliminate these repetitions and improve algorithm performance. The suggested technique uses tree data structure instead of linear array.

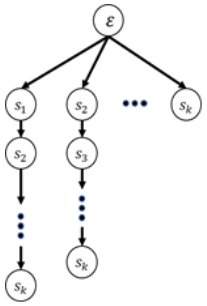

Fig. 1. Tree created for episode *S*

The extracted possibilities are passed on to the final step for the calculation of probabilities which will later be used to predict future user actions. The remaining steps are reproduced from [8] without alteration.

2.1. All Possible Context Extraction

This research proposes a faster and more efficient technique to create and store possibility contexts in the SPEED algorithm. A prefix tree (also known as trie tree) design is introduced in the process that can effectively conserve memory and serve as a container for possibilities generated from episodes [15]. The possibility tree contains a root node ε and multiple subtrees connected to ε using singly connected linked lists. Within the tree, node T_x occurring as a child of node T_y represents that, event T_y is preceded by event T_x in the training dataset.

For each episode s of length k, k number of subtrees are formed under root node ε. Within each subtree, every event s_n is added to its preceding node $s_{(n-1)}$ as a child and assigned a frequency of value 1. In the event of a repeating subtree, the node will preexist in the structure. In that case, the frequency of that node is incremented by 1. The general structure of the tree for episode s is shown in Fig. 1.

To make the tree generation step faster, a linear array *nodes_array* is appointed for each new episode to hold the pointers to the leaf nodes. For every event s_n in episode s, s_n is added as a child to all the pointed nodes in the *nodes_array*. Finally, the node s_n is itself added to the *nodes_array* as a new node. This step eliminated the need to repeatedly traversing the tree in order to add nodes to their respected positions. The successive 4 iterations of the algorithm for episode 'Abca' is depicted in Fig. 2.

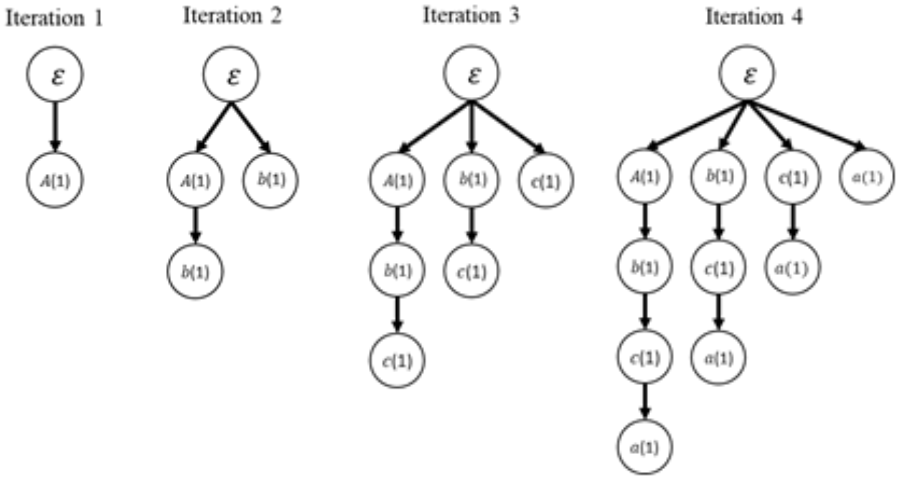

Fig. 2. Steps of the algorithm to form possibility tree for episode 'Abca'

During first iteration, the event 'A' is added to the tree and concurrently pushed into *nodes_array*. On second iteration, the event 'b' is first added to the tree. Then it is added as a child to all the nodes residing in *nodes_array*. In this case, *nodes_array* contains 'A', therefore, a child 'b' is added to it. Similarly, 'c' is added to all leaf nodes in iteration 3, and event 'a' is added in the final iteration. In this specific example, the nodes are created for the first time, hence, value 1 is assigned as their frequencies. If the node already existed, the frequency would have been incremented by 1.

To establish the run-time memory effectiveness of the proposed improvements, we have presented following 4 hypotheses in this paper.

Hypothesis 1: Worst case space complexity of the possibility function of SPEED algorithm for any episode of length k is in the order of k to the power of three.

Proof: After the episode generation phase, all possible contexts sharing adjacent events are considered from the episodes and are stored in an array. For any episode s of length k, the number of possibilities of length t will be k-t+1.

To demonstrate this, let us consider an episode 'Abca'.

Jurnal Informatika
Vol. 15, No. 2, May 2021, pp. 115-122

Possibilities containing 1 letter are 'A', 'b', 'c' and 'a' (in total 4).
Possibilities containing 2 letters are 'Ab', 'bc' and 'ca' (in total 3).
Possibilities containing 3 letters are 'Abc' and 'bca' (in total 2).
Possibility containing 4 letters is 'Abcd' (in total 1).

Thus, we can infer, in SPEED algorithm, the total number of memory required (letters) for episode of length k,

$$M_1(k) = \sum_{t=1}^{k} t * (k - t + 1) = (k + 1) * \sum_{t=1}^{k} t - \sum_{t=1}^{k} t^2 = \frac{k*(k+1)^2}{2} - \frac{k*(k+1)*(2k+1)}{6} = \frac{k^3+3k^2+2k}{6} \quad (1)$$

Hence, the worst-case space complexity is in the order of O(k^3), where k is the number of events in an episode.

Hypothesis 2: Worst case appears when a large episode covers the whole data and there is no repetition of events.

Proof: If an event sequence D contains n number of episodes of length k_n, where k=∑k_n , then we know (2a) and (2b),

$$k^3 \geq \sum(k_n)^3 \quad (2a)$$

$$\text{And } k^2 \geq \sum(k_n)^2 \quad (2b)$$

Therefore, we can conclude from Equation (1)

$$M_1(k) \geq M_1(k_n) \quad (3)$$

That means, when the data sequence is divided into two or more episodes, the allocated memory can not be larger than the whole dataset being a single episode (3).

Hypothesis 3: Worst case space complexity of the suggested possibility function for any episode of length k is in the order of k to the power of two.

Proof: The prefix tree implemented in this research connects the nodes using singly connected linked lists. Every node contains one letter (to denote its event), frequency, and an address to the next node. For an episode s of length k, the number of nodes at level l will be k-l+1, where 1≤l≤k.

This argument can be demonstrated by visualizing the tree formed by the episode 'Abca' in Fig. 3. Here, for every level, the node count is decremented by 1.

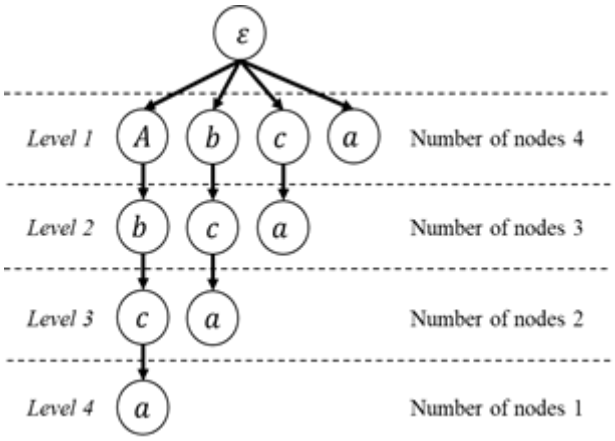

Fig. 3. Context tree formed for episode 'Abcd'

Hence, the total amount of required memory for episode of length k is:

$$M_2(k) = \sum_{l=1}^{k}(k - l + 1) = \frac{k*(k+1)}{2} \quad (4)$$

Therefore, the worst-case space complexity of the proposed possibility generation module is in the order of $O(k^2)$, where k is the number of events in an episode.

Hypothesis 4: Worst case appears when a large episode covers the whole data and there is no repetition of events.

Proof: Similar to Hypothesis 2, when a data sequence is divided into n number of episodes of length k_n, where $k=\sum k_n$, we can say,

$$k^2 \geq \sum(k_n)^2 \tag{2b}$$

Putting these values in Equation (4) we find,

$$M_2(k) \geq M_2(k_n) \tag{5}$$

This proves that dataset having one large episode covering the whole sequence will require maximum amount of memory.

In the previous SPEED algorithms, the Tree generation step following the All possible context generation module was used to simultaneously create the decision tree and allocate the probability of outcomes. This resulted in misaligned probabilities among different branches of the tree. This research separates the tree generation from the probability allocation. This technique can ensure uniform distribution of probabilities and significantly improve the accuracy of prediction.

The decision tree formed during the Possibility generation phase is traversed using a Depth First Search (DFS) traversal to visit all nodes and set their respective probabilities. At any given level the algorithm first goes to the leftmost child node and recursively reaches its bottom. Upon reaching a leaf node, sets its probability value and travels back to its immediate parent node. When the leftmost node child node is already traveled, the algorithm steps to the child node to its right and follows the same recursive procedure. This way every subtree is visited from bottom to top and left to right. This process can result in having better prediction accuracy than the SPEED algorithm.

3. Results and discussion

In this section, proposed algorithm is compared with SPEED in terms of memory overhead and runtime in order to validate the presented research. MavHome dataset is used in the simulations of both approaches. The training dataset contains 1675 chronological events.

Firstly, the memory consumption of the two algorithms in Possibility module are calculated and contrasted. To get a reliable estimation, the actual byte count of events is considered, overlooking the memory required for frequency and time. It is to be noted that, in terms of SPEED, the individual events in every possibility allocates 1 byte of memory. So, for a possibility Abc, 3 bytes are occupied in the memory. Fig. 4(a) exhibits the average memory required for different episode lengths. The use of array in SPEED results in an exponential growth of memory as the episode length increases. On the other hand, a linear growth is observed in the modified approach, as it utilizes a tree data structure. The final result can be viewed in Fig. 4(b) which shows the cumulative growth of allocated memory for successive episodes. Due to the exponential growth in SPEED, the resultant operating memory increases rapidly. This is obvious that for larger datasets, the algorithm performance will decline significantly and for even larger datasets, this approach will fail. The proposed algorithm can resolve this issue by effectively utilizing runtime memory. The simulations result shows an improvement of cumulative memory efficiency by nearly 66.69%.

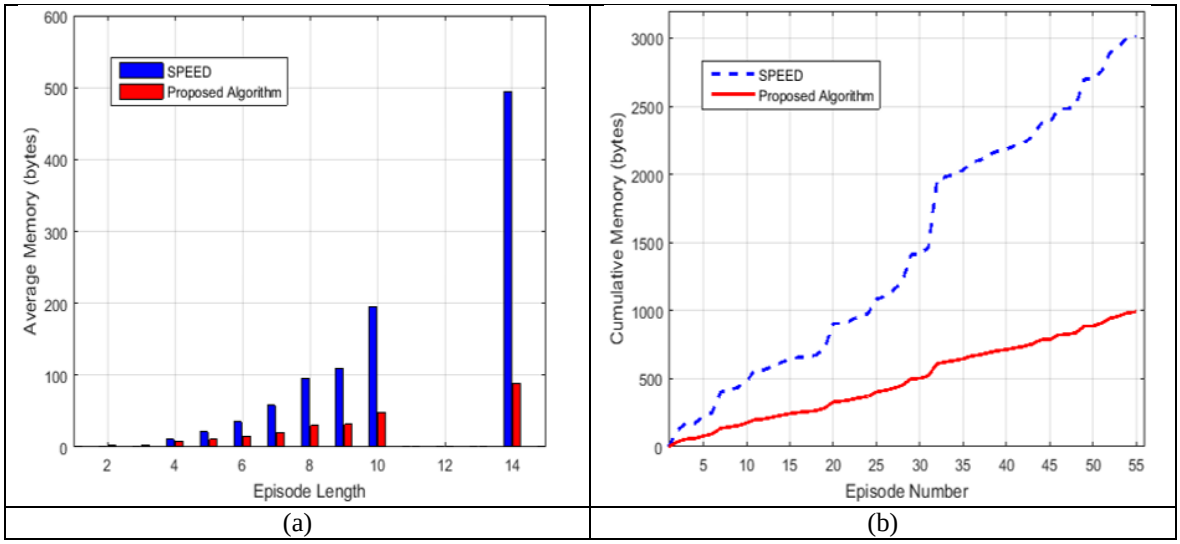

Fig. 4. (a) Average memory comparison between SPEED and this research, (b) The cumulative growth of allocated memory for successive episodes

Secondly, running time of the possibility generation module is calculated for both algorithms with respect to episode lengths. The performance results of the two algorithms on MavHome data is demonstrated in Fig 5. During the timing phase, utility function calls are ignored in order to obtain authentic results which can assist in a better comparison between the two procedures.

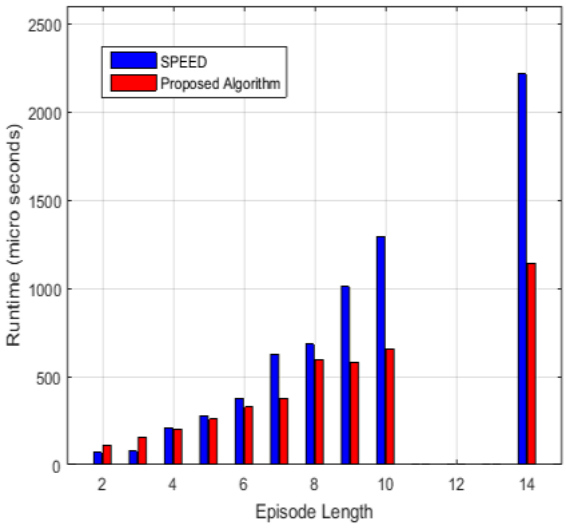

Fig. 5. Running time comparison between SPEED and this research

Fig. 6 charts the accuracy of the proposed algorithm and M-SPEED on MavLab dataset with respect to episode lengths. Here, the episode lengths refer to the number of sequential events constituting a particular activity. An accuracy of 78% for episode length 4 indicates that, when the trained model predicts activity after 3 consecutive known events, an average of 78% accuracy is achieved. The graph shows a uniform increase in prediction capability of around 10% for the presented approach. This is a concrete establishment that the proposed algorithm performs better than the previous M-SPEED in every possible episode length.

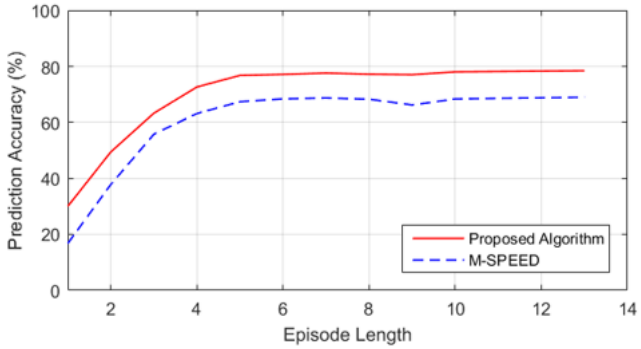

Fig. 6. Accuracy comparison of SPADE and M-SPEED on MavLab dataset with time verification

Linear searching of matching possibilities in the SPEED algorithm causes it to greatly depend on the data composition. For example, even for an episode of length 1, the algorithm needs to search the whole possibility array to increase its frequency. On the other hand, due to guaranteed addition of nodes to the tree, the Possibility module presented in this research does not require any searching. Thus, a gradual increase is perceived in the corresponding graphs. Overall, the average runtime experiences nearly 37% improvement in the new algorithm.

This study shows that, modifying the possibility generation function in the SPEED algorithm according to the proposed technique can provide a solution to the memory consumption issue, while at the same time improving the runtime of the algorithm. The results agree with the hypothesis and prove the effectiveness of the presented technique.

4. Conclusion

In the fields of automation for user convenience, M-SPEED delivers a significant contribution by providing highly accurate user activity predictions. But exponential memory allocation and extended runtime cause this algorithm to suffer greatly when large datasets are considered. Live data from existing smart homes comprises enormous amount of data instances and thus, M-SPEED becomes ineffective in various occasions. This research demonstrates an approach to substitute linear array with tree data structure in the all possible context generation phase, in order to make M-SPEED suitable for larger datasets and real-world situations. The proposed algorithm also improves the prediction accuracy for smart homes. This study will assist future researches on activity prediction to effortlessly integrate massive data sources.

Acknowledgment

This research is financially supported by the Universiti Kebangsaan Malaysia, Malaysia. Project code: DIP-2017-003.

Declarations

Author contribution. The author read and approved the final paper.
Funding statement. None of the authors have received any funding or grants from any institution or funding body for the research.
Conflict of interest. The author declares no conflict of interest.
Additional information. No additional information is available for this paper.

References

[1] "Washington, D.C. : World Bank Group. World report on disability : Main report (English)," 2011. [Online]. Available: http://documents.worldbank.org/curated/en/665131468331271288/Main-report.

[2] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali, "A Review of Smart Homes—Past, Present, and Future," *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 42, no. 6, pp. 1190–1203, Nov. 2012, doi: 10.1109/TSMCC.2012.2189204.

[3] S. Wu *et al.*, "Survey on Prediction Algorithms in Smart Homes," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 636–644, Jun. 2017, doi: 10.1109/JIOT.2017.2668061.

[4] A. Bhattacharya and S. K. Das, "LeZi-Update: An Information-Theoretic Framework for Personal Mobility Tracking in PCS Networks," *Wirel. Networks*, vol. 8, pp. 121–135, 2002, doi: 10.1023/A:1013759724438.

[5] K. GOPALRATNAM and D. J. COOK, "ACTIVE LEZI: AN INCREMENTAL PARSING ALGORITHM FOR SEQUENTIAL PREDICTION," *Int. J. Artif. Intell. Tools*, vol. 13, no. 04, pp. 917–929, Dec. 2004, doi: 10.1142/S0218213004001892.

[6] K. Gopalratnam and D. Cook, "Online Sequential Prediction via Incremental Parsing: The Active LeZi Algorithm," *IEEE Intell. Syst.*, vol. 22, no. 1, pp. 52–58, Jan. 2007, doi: 10.1109/MIS.2007.15.

[7] V. Jakkula and D. J. Cook, "Mining sensor data in smart environment for temporal activity prediction," *Poster Sess. ACM SIGKDD, San Jose, CA*, 2007. Available: Google Scholar.

[8] M. R. Alam, M. B. I. Reaz, and M. A. Mohd Ali, "SPEED: An Inhabitant Activity Prediction Algorithm for Smart Homes," *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 42, no. 4, pp. 985–990, Jul. 2012, doi: 10.1109/TSMCA.2011.2173568.

[9] M. Marufuzzaman, M. B. I. Reaz, M. A. M. Ali, and L. F. Rahman, "A Time Series Based Sequence Prediction Algorithm to Detect Activities of Daily Living in Smart Home," *Methods Inf. Med.*, vol. 54, no. 03, pp. 262–270, Jan. 2015, doi: 10.3414/ME14-01-0061.

[10] M. Marufuzzaman, M. B. I. Reaz, L. F. Rahman, and A. Farayez, "A Location Based Sequence Prediction Algorithm for Determining Next Activity in Smart Home," *J. Eng. Sci. Technol. Rev.*, vol. 10, no. 2, pp. 161–165, Jun. 2017, doi: 10.25103/jestr.102.19.

[11] A. Farayez, M. B. I. Reaz, and N. Arsad, "SPADE: Activity Prediction in Smart Homes Using Prefix Tree Based Context Generation," *IEEE Access*, vol. 7, pp. 5492–5501, 2019, doi: 10.1109/ACCESS.2018.2888923.

[12] W. J. Teahan, "Probability estimation for PPM," *Proc. NZCSRSC'95*, 1995. Available: Google Scholar.

[13] D. J. Cook and M. Schmitter-Edgecombe, "Assessing the Quality of Activities in a Smart Environment," *Methods Inf. Med.*, vol. 48, no. 05, pp. 480–485, Jan. 2009, doi: 10.3414/ME0592.

[14] H. Alemdar, H. Ertan, O. D. Incel, and C. Ersoy, "ARAS human activity datasets in multiple homes with multiple residents," *2013 7th Int. Conf. Pervasive Comput. Technol. Healthc. Work.*, pp. 232–235, 2013. Available: Google Scholar.

[15] R. Begleiter, R. El-Yaniv, and G. Yona, "On Prediction Using Variable Order Markov Models," *J. Artif. Intell. Res.*, vol. 22, pp. 385–421, Dec. 2004, doi: 10.1613/jair.1491.