IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 27, 2020, accepted December 28, 2020, date of publication December 31, 2020,
date of current version January 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048423

Device Interaction Graph: Directed Decision Graph for Settings Auto-Completion

RAMASAMY KANNAN, (Senior Member, IEEE), SAI HARISH PATHURI, AND DEEPAK KUMAR
Samsung Research and Development Institute India - Bengaluru, Bengaluru 560037, India
Corresponding author: Ramasamy Kannan (ram.kannan@samsung.com)

ABSTRACT Knowledge generated from appliances usage data can be used for personalizing and reducing the number of user interactions. Appliances such as Washing Machines (WM) are capable of storing the settings data selected by the user, during operation. Before the operation of a WM, the user has to enable the options for these settings for a preferred mode of operation. We have proposed a Device Interaction Graph solution that automatically recommends and auto-completes the user's preferred settings in real time. For direct manual interactions on the WM, the auto-completion solution is applied when the user incrementally changes the settings on the user interface. For voice based command interactions with the WM, the auto-completion solution would be applied after understanding any incomplete settings requirements mentioned by the user. This settings auto-completion solution improves the user experience by reducing the number of manual or verbal interactions required to enable the operation of the device. We have applied the auto-completion solution on the data obtained from 158,213 connected WMs and achieved recommendation accuracy of 89.79% with an average 3.132 interactions.

INDEX TERMS Data analysis, decision theory, knowledge based systems, knowledge representation, home automation, Internet of Things, smart homes, tree graphs, washing machines.

I. INTRODUCTION

With the adoption of Internet of Things (IoT) in Smart homes [1], all devices and appliances such as Washing Machines (WM) are now connected to the cloud. The analysis of the data generated from IoT enabled Smart Homes has brought various benefits for remote monitoring/control, reduced energy consumption and improved appliance usage experiences [1]–[3]. In this paper we will evaluate one such approach to improve user experience while using connected appliances such as WM. Washing Machines have multiple settings which the users have to select before enabling their operation. In this paper, we have proposed a settings auto-completion solution using our proposed Device Interaction Graph (DIG). Many recently released WM models are Internet of Things (IoT) enabled and the data from these WMs can be analyzed on the Cloud. This analysis can improve the user experience while using WM. We have applied the DIG solution individually on the data generated by 158,213 WMs. Based on this analysis, the new IoT enabled WMs, which have more on-device memory and processing capabilities, can store and process all the previously selected settings and options values. The DIG auto-completion algorithm can be generated on the cloud and loaded on the WM. This will enable generation of on-device recommendations. Alternatively, for the new IoT enabled WMs that have lower memory and processing capabilities, the WM settings will be applied from the cloud. The DIG auto-completion algorithm will be applied to the current input settings on the cloud and the output recommendations will be sent back to the device.

The User Interface (UI) design strategies for intelligent WMs are discussed in [4]. For enabling manual WM operation, the user has to select multiple settings on the UI as shown in Fig. 1. The DIG understands the WM usage patterns and reduces the number of user interactions required to enable the operation. In this paper, we have proposed a settings auto-completion solution that understands the past user-WM usage patterns and reduces the number of interactions required to enable WM operation. The proposed DIG transforms all the historical WM usage settings data into a directed acyclic decision graph representation. With each setting and option selected by the user, the auto-completion algorithm keeps updating possible options for the remaining settings the user is most likely to use.

For WMs that support voice based human interactions, the user will give information for the WM settings to be

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

FIGURE 1. An example washing machine user interface for selecting preferred mode of operation. The user has to manually select multiple options before the washing is enabled. Our auto-completion solution will reduce the number of manual operations on the WM.

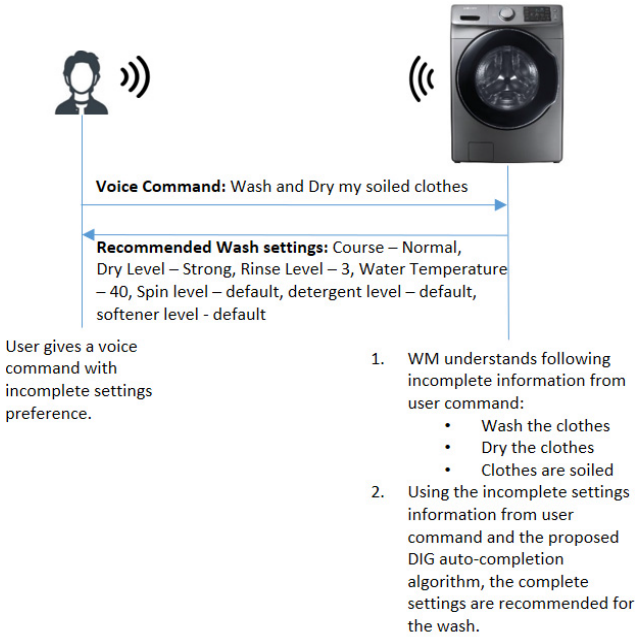

FIGURE 2. An example use-case showing the benefits of applying the proposed auto-completion solution for voice based commands from the user to the WM. The complete WM settings are recommended combining the information from the user's voice command and DIG representation.

enabled, through the voice command. The Artificial Intelligence (AI) voice assistant available (integrated) in the WM will decode the command and output the WM settings mentioned by the user. The AI voice assistant is out of scope of this paper, and existing research [5] can be referred on this topic. We have assumed that the user's intended WM settings are decoded and available as input to our proposed solution. While there are many settings, the user provides input (voice command) on certain wash preference and this has to be mapped to the values for the existing WM settings. An example voice command and its mapping to existing settings is shown in Fig. 2. The goal of this paper is to ensure efficient auto-completion of the preferred settings based on the partial inputs received from the voice commands. The DIG auto-completion algorithm, takes in the available inputs and recommends the preferred WM settings. To enable the WM user's preferred mode of operation using incomplete settings, the DIG representation model efficiently stores the WM settings enabled during previous operations.

II. BACKGROUND AND RELATED WORK

There are limited related works for improving User-WM interactions using the generated WM data. We have discussed and compared the relevant WM related work in this section. A Neural Network (NN) based fuzzy controller to reduce water and energy consumption during WM operation, using historical data is explained in [6]. Solutions that use a combination of fuzzy controller, NN and genetic algorithms to control WM operation are discussed in [7], [8]. Solutions for estimating the weight of clothes using Machine Learning (ML) approaches are discussed in [9], [10]. In this paper, we have proposed a WM settings auto-completion solution using a novel knowledge based system. Our solution does not use NNs or fuzzy controllers or genetic algorithms. These solutions are used for providing the top recommendation. We need a solution that uses the settings already selected by the user, to recommend the remaining settings. Another solution for estimating the cleanliness of the clothes using inputs such as amount of washing powder, duration of the wash and washing temperature is explained in [11]. A ML based approach for classifying the fabrics inside the WM is described in [12]. Another ML based approach for estimating the moisture of clothes loaded in WM is discussed in [13]. Based on the survey of existing literature on WM, there is currently no existing solution that can be used for settings auto-completion during real-time user interactions.

Next we look at the existing literature that deal with similar knowledge based systems such as Decision Trees and Directed Graphs. The proposed solution recommends the most probable options for the remaining settings, based on the limited settings that has already been selected. This is equivalent to reducing the uncertainty in data for the reminder of the settings, after the user has already selected one or more device setting options. In Information theory, Entropy [14] is a measure of uncertainty in the data. One of the important aspects of our proposed approach, is to reduce this uncertainty. Let us consider the case where there are two device settings 'A' and 'B', and the setting 'A' is used often, while setting 'B' is rarely used. This results in the entropy (or measure of uncertainty) of setting 'A' being higher than that of setting 'B'. The use of data mining for data generated by IoT enabled devices, is explained in [15]. The processing of a sequence of data events in data mining, irrespective of the time stamps of their occurrence are discussed in various literature [16]–[18]. The proposed DIG solution also

processes the events related to WM setting irrespective of the selection order. But in our solution we have used a novel directed decision graph approach compared to these existing data mining solutions. Various Tree based algorithms have been used for knowledge representation and inference [19]. Algorithms such as Decision Trees [20] and Random Forest [34] can exploit the different levels of uncertainty in each of the device settings that are selected to provide automatic recommendations. Traditional Decision Tree algorithms are used to split the input data continuously in order to maximize a prediction outcome. The main reason a traditional Decision Tree ML algorithm cannot be used for the problem we have considered, is because: a) Relationships between the nodes cannot be added on a decision tree and hence a decision graph is required; b) The settings themselves are to be used as both the input and the target features; and c) A customized graph design (that cannot be supported on decision trees) is required for the purpose of settings auto-completion. We have proposed the DIG representation and an algorithm to overcome this issue. A directed graph to make decisions based on time-evolving event sequences is discussed in [21]. This directed graph implementation in [21] does not store the event sequences that are repeated over time. In our approach, the DIG representation has ensured that the repetition of event sequences are captured. The proposed DIG has combined the benefits of decision trees and directed graphs to generate an optimal representation for device usage data. The DIG has used the principles of induction similar to decision trees [20]. This has ensured that the settings with highest uncertainty are placed nearer to the root and the settings with lowest uncertainty are placed closer to the leaves. The traversal behavior of the DIG representation is similar to traversing a TRIE data structure [22] while the settings are recommended. This is because traversal of TRIE tree gives the complete sequence of the possible text from beginning to the end. There is also a Compact Decision Tree algorithm [23] that can be used to predict the next element in a sequence of data. While this approach specifically considers representing data sequences in a Tree structure, in our case of device settings selecting the data cannot be represented as fixed sequence. The order of settings represented in the DIG will change based on the uncertainty in the data. Storing and classifying topologically structured knowledge in the form of decision trees is explained in [24]. In our case the settings data cannot be structured topologically in order to solve the auto-completion problem. A Decision Tree variant to work on unknown data range/boundaries is discussed in [25]. In the scenario that we have considered, the WM setting values are bound within predefined ranges. Hence there will be no uncertainty in the bounds of the input data. The generation of Decision Trees to work on dynamic and streaming data is discussed in [26]. In the WM case, there will be no streaming data, as we generate DIG using only the past WM usage settings that are already stored in memory.

We have compared the proposed solution with data mining approaches, Machine Learning (ML) and Deep Learning (DL) approaches in the section on Experiments. The use of big data processing in combination with ML/DL models, to predict outcomes involving large number of connected IoT devices is discussed in [27], [28]. There are well known data mining techniques to extract knowledge from large volumes of data (big data), generated by IoT devices [29]. A survey of solutions for extracting knowledge from IoT data using Frequent Pattern Mining approaches is discussed in [15]. The ML and DL models are developed to extract additional knowledge from existing structured data. A survey of the various issues faced while using IoT big data when applying ML algorithms and some of the existing solutions are discussed in [31]. DL algorithms [32] are showing huge potential in learning hidden representations from large volumes of data. A study of various DL models and frameworks used on IoT Big Data along with the challenges faced are discussed in [33]. In this paper, we have used Big Data processing to convert WM logs to wash sessions. Using these wash sessions we have generated DIG representations for each WM. We have compared our proposed DIG representations with recommendations made from pattern mining, ML and DL approaches. For the case of ML algorithm, we have compared with recommendation made by Random Forest algorithm [34]. The Random Forest Algorithm uses a collection of decision trees to predict the outcome. For the case of DL algorithms, we have considered Fully Connected Neural Network (FCNN) [27], [28], [35] and Long Short Term Memory (LSTM) sequence model [36].

The major contributions of this paper are:

1) Algorithm for generation of a Device Interaction Graph (DIG) representation that enable auto-completion of settings for Washing Machines.

2) Algorithm for traversal of the DIG representation to recommend settings during User-WM interactions;

The rest of the paper has been structured as follows: Section III provides information about the WM data and relevant insights for the auto-completion problem. Section IV describes the various steps in generating the proposed DIG representation. Section V explains the benefits of the proposed algorithm with results. The results have also been compared with existing Top-K data mining, ML and DL approaches. Section VI concludes this paper by highlighting the important achievements in this paper and future work scope in this area.

III. WASHING MACHINE DATA AND INSIGHTS

The IoT enabled WMs are connected to cloud, enabling their data to be analyzed for providing a better user experience. The data generated from 158,213 IoT enabled WMs are processed and analyzed on the cloud. The processing of the WM data using cloud infrastructure is similar to the methods we have applied for IoT enabled Air Conditioners in [27], [28]. The event logs from the WM were sent to the cloud using IoT infrastructure. The event logs on the cloud were converted into tables. The columns which were useful for our current solution were retained in the Device Settings Table (DST).

TABLE 1. Settings and options for all WM considered in this paper.

Settings	Options
Wash Course	15 Quick Wash, 15' Express, 59 Wash+Dry, Active Wear, Air Bedding Care, Air Refresh/Sanitize, AIR Wash, Allergen, Auto Optimal Wash, Baby CareBedding, Bedding/Waterproof, Colors/Darks, Cotton, Cotton Dry, Daily Wash, Dark Garment, Deep Clean, Deep Steam, Deep Wash, Delicates, Drain/Spin, Drying, Drying Drum, Eco Cold, Eco Drum Clean, Eco Drum Clean+, Eco Tub, Clean, Energy Saving, Heavy Duty, Hygiene Steam, Normal, Outdoor, Outdoor Care, Outdoor Refresh, Padding Care, Perm Press, Power Bubble, Quick Dry, Quick, Wash, Rinse+Spin, Sanitize, Self Clean, Self Clean+, Soak+Normal, Speed Wash+Dry, Spin, Super Clean, Super, Eco Wash, Super Speed, Synthetics, Synthetics Dry, Towels, Wash&Wear, Whites, Wool, Wool/Lingerie
Soil Level	ExtraHeavy, Heavy, Normal, Light, ExtraLight, None
Rinse Cycles	0, 1, 2, 3, 4, 5
Spin level	RinseHold, NoSpin, 400, 800, 1000, 1200, 1400
Dry level	Very Strong, Strong, Normal, Low, None
Water Temperature	None, Cold, 20, 30, 40, 60, 90
Detergent level	0, 1, 2, 3
Softener level	0, 1, 2, 3

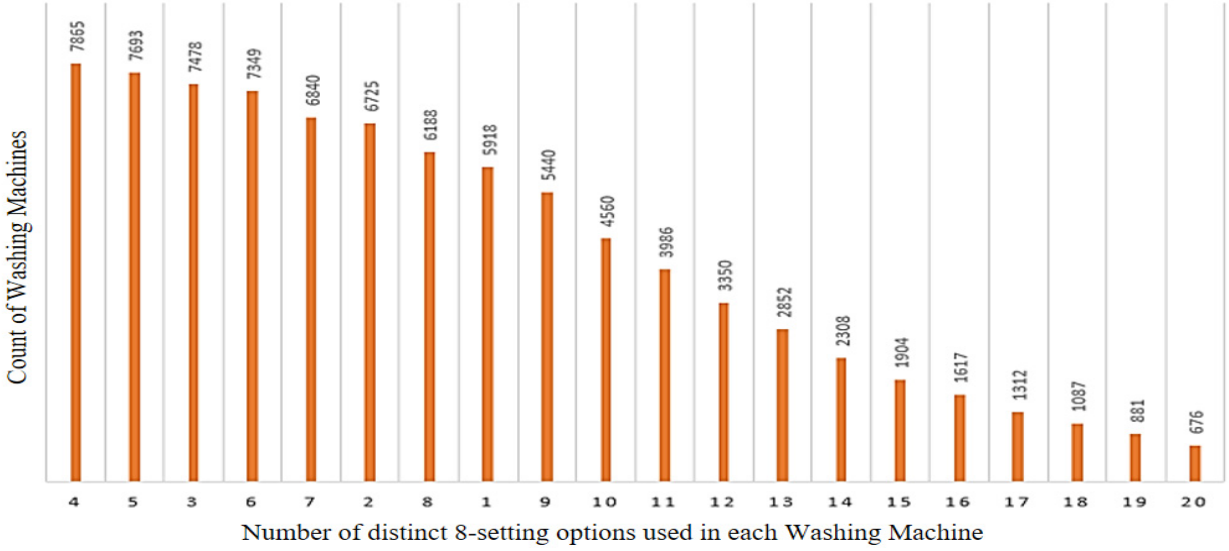

FIGURE 3. Count of WM user (vs) No. of distinct 8-setting options selected during 6 months of usage. Even though there are large number of options available for each WM setting, each user has a preferred combination of the 8-settings which are repeated often in all the washes.

The columns in the DST consisted of WM settings data along with the timestamps that the settings events were received. As shown in Table 1, the major WM settings data consisted mainly of the 8 settings such as 'Course', 'Rinse Level', 'Spin Level', 'Dry Level', 'Soil Level', 'Water Temperature', 'Detergent Level' and 'Softener Level'. The 'Options' column shows the various options that were available under each of the settings. These settings and their options are available for selection by the user on the WM before each wash. For the set of eight WM settings represented as {'Wash Course', 'Soil Level', 'Rinse Level', 'Spin Level', 'Dry Level', 'Water Temperature', 'Detergent Level', 'Softener Level'}, an example for the options selected would be {'Cotton', 'Heavy', '2', '1000', 'Strong', '30', '2', '2'}.

We have considered processed data from 158,213 WMs over a period of 6 months having over 10.5 million washes, to provide certain insights regarding the users wash behavior. Fig. 3 shows the histogram for count of distinct settings combinations selected on all WMs during the 6 month period. The x-axis in Fig. 3 has been truncated to show the Top-20 settings combination counts selected by the users. This distribution shows that, even though there are large number of options available for each WM setting, each user has a preferred combination of the 8-settings which are repeated often in all the washes. A maximum of 7865 WMs out of 158,213 WMs have used 4 distinct settings combination during the 6 month period. Currently, other than time of the wash and the settings selected during these washes, there are no other input features in the processed data that can be used for auto-completion of the preferred wash. Therefore we have only used each user's preferred combination of settings and options in designing the auto-completion algorithm.

IV. DEVICE INTERACTION GRAPH

For enabling auto-completion of the settings, an efficient representation of the past WM data is required for minimal

TABLE 2. Example device preference table with unique '8-Settings Options' and probabilities for one WM.

Course	Soil Level	Spin Level	Rinse Cycles	Dry Level	Detergent Level	Softener Level	Water Temperature	Probability
Super Speed	1	1400	2	2	1	1	40	0.409
Super Speed	1	1400	2	2	1	1	60	0.045
Cotton	1	800	2	2	1	1	40	0.364
Cotton	1	800	2	2	1	1	60	0.091
Cotton	1	1400	2	2	1	1	40	0.045
Cotton	1	1400	2	2	1	1	60	0.045

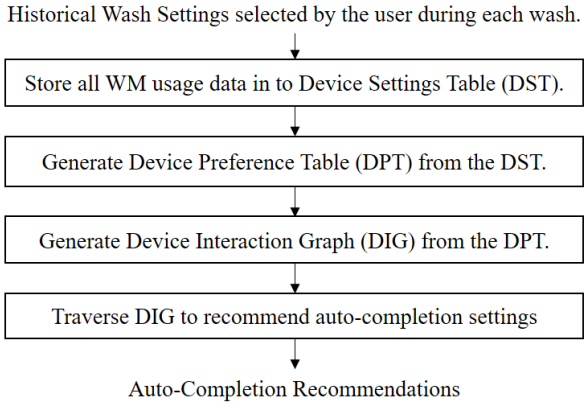

FIGURE 4. Overall steps for enabling auto-completion of WM settings.

interactions and faster lookup time. The steps for converting the WM usage data into the DIG representation are shown in Fig. 4. The past settings data selected during each wash on each WM is stored in Device Settings Table (DST). First, a Device Preference Table (DPT) is generated from the DST. Next, the DPT is used to generate a DIG representation of the same data. The DIG is a directed acyclic decision graph representation. The DIG representation is designed keeping in mind WM auto-completion requirement, and is better suited for querying incremental information compared to the DPT representation. Finally, The DIG traversal algorithm ensures that the most probable settings options are recommended on the WM continuously, based on the partial selection by the user.

A. DEVICE PREFERENCE TABLE GENERATION

The flow chart shown in Fig. 5 explains the generation of DPT by using the Device Settings Table (DST). DST consists of all past wash settings for each wash along with the wash time in each row. The DST and DPT are unique for the data from each WM. There are multiple rows in DST with the same 8 wash settings which are repeated more than once due to the frequent repeated wash (as shown in Fig. 1). These multiple rows in the DST are converted to a single row on the DPT. In order to do this, first, the 8 setting values in each row are concatenated in a specific ordered sequence. The concatenated sequences are then stored in a new column named WSC in DST. Next, the count of distinct WSC values in DST are computed. Then TWC is computed by counting of all washes in DST. The DPT generated has unique 8-setting options in each row and its usage probability. The probability in each row gives a measure of how often each '8-settings options' are repeated by the user. The '8-setting options' with the highest probability in the DPT is the most preferred settings options of the WM. An example DPT with unique '8-setting options' and their probabilities are shown in Table 2. The DPT shows the usage for a single WM during a 6-month period. The 'Course' setting has two distinct options of 'Super Speed' and 'Cotton'. The 'Spin Level' setting has two distinct options of '800' and '1400'. The 'Water Temperature' setting has two distinct options of '40' and '60'. The other 5 settings in the table given by 'Soil Level', 'Rinse Cycles', 'Dry Level', 'Detergent Level' and 'Softener Level' have only 1 distinct option used. Each row in the DPT is a distinct combination of options for the 8 WM settings. The 'Probability' column denotes the usage probability of each settings combination. The distribution of the probability shows that, the WM user prefers certain combination of settings compared to others. Compared to the multiple available options as shown in Table 1, the user has used only a limited set of options available on the WM model for the various settings. In the next section, we explain how the DPT table is used to generate the DIG representation.

B. GENERATION OF DEVICE INTERACTION GRAPH

The Device Interaction Graph (DIG) representation is an optimal representation of the DPT. The DIG representation is used later to implement the auto-completion solution. Algorithm 1 describes the steps for converting a DPT to DIG representation. The DIG algorithm is a recursive algorithm that identifies the settings with the highest to the lowest entropy [14]. The settings which are frequently changed have higher entropy compared to those which are less frequently changed. A setting which has not been changed has an entropy of 0. The setting with the highest entropy is placed nearest to the root and the settings with zero entropy are placed in the leaves of the decision graph. Once the setting/column with highest entropy is identified, each distinct option for the setting is inserted to the DIG node as an edge. Once the edge is added, the rows having the setting option are filtered and removed from the DPT. Once all the edges for each setting option are added the setting column is removed from the DPT. The Information Gain is the change in entropy of the DPT before and after the removal of the

Algorithm 1 DIG DEVICE_INTERACTION_GRAPH(DPT)

Input: DPT - Device Preference Table
Output: DIG - Global memory for Device Interaction Graph (initially empty)

```
Create new DPT column S_v by concatenating values in 8 settings columns
Create list S_N from DPT by storing the names of the columns
IF (DPT != empty):
  Calculate Entropy with respect to S_v as defined in (1)
  IF (Entropy != 0):
    Calculate Information Gain for all settings in S_N as defined in (2)
    Create a new node as the setting name S_n, the setting in S_N having highest Information Gain.
    FOREACH distinct option k of S_n in DPT:
      Filter out and remove the rows containing option k for S_n and create new table New_DPT
      S_{n+1} = DEVICE_INTERACTION_GRAPH(New_DPT)
      Add node S_{n+1} as child to node S_n.
      Add relation R(S_n, S_{n+1}) between node S_n and S_{n+1} using equation (3)
    END FOR
  ELSEIF (Entropy == 0)
    Create leaf node in DIG with the remaining settings and options in S_v.
  END IF
ELSE:
  RETURN DIG
END IF
```

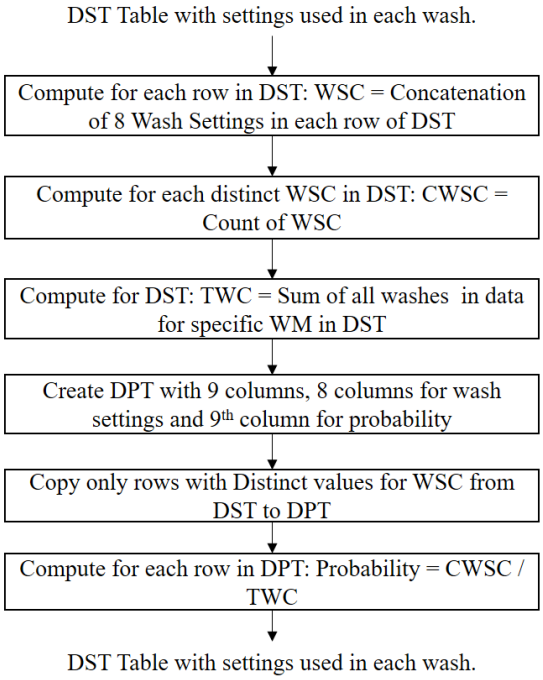

FIGURE 5. Steps for generating device preference table from device settings table.

column/setting. The setting with the maximum Information Gain is recursively identified and removed from the DPT and placed as new nodes on the DIT. Entropy (*Ent*) and Information Gain (*I.G.*) are defined in [14] and are adopted for DIG as shown in (1) and (2) respectively. For DIG there is no target feature, as the most probable 8-settings have to be recommended from the past settings data used. In order to compute the entropy of each setting and to identify the setting with maximum information gain, a derived target feature is required. For this purpose, S_v denoting the concatenation of all the 8-setting options is computed as the target feature (v denotes the iterator to go over each option in S_v). S_n used in the pseudo code denotes the setting with the highest entropy in DPT for that iteration.

$$Ent(DPT) = \sum_{each(V)} (p\,(S_v) * log(p\,(S_v))) \tag{1}$$

$$I.G. = [Ent(DPT)]_{before\_filter} - [Ent(DPT)]_{after\_filter} \tag{2}$$

During the construction of DIG, the setting with the highest information gain is identified. The name of this setting becomes the new node value. For each option v in the setting S_n, a separate subgraph with sub-nodes have to be generated. So all the rows with option value v in setting S_n column in DPT are filtered out and removed to generate a new table New_DPT. Once there are no more rows and all options for the setting S_n are filtered out, the column is removed from the table. The column count of the New_DPT table reduces by 1. This New_DPT table is then used to generate the subgraphs for the current node S_n. The Device_Interaction_Graph() function (shown in Algorithm 1) is called recursively till the entire graph is generated. The DIG representation is a Directed Acyclic Decision Graph having edges denoting the relationship between nodes. Once the child node for the DIG is identified as shown in Algorithm 1, the relationships between the parent node and child node has to be added. The relationships in DIG, plays an important role in deciding the preferred settings path during each user update on the WM.

Algorithm 2 TRAVERSE_DIG(S_{n+1}, S_n.option)

Inputs: S_{n+1} - DIG Node at level $n+1$, S_n.option – Option selected by user for Setting S_n at level n
Outputs: None

1. IF (S_n.option belongs to DIG):
2. Disable paths having $R(S_n, S_{n+1})$ with setting option != S_n.option
3. ENDIF
4. IF (S_n != leaf node):
5. IF ($R(S_n, S_{n+1})$ with: MAX($p(S_n.\text{option}/S_{n-1}.\text{option})$) (or) MAX($p(S_1.\text{option})$)):
6. Recommend option for S_n: MAX($p(S_n.\text{option}/S_{n-1}.\text{option})$) for setting S_n (or) MAX($p(S_1.\text{option})$) for S_1
7. S_{n+1} = next node S_{n+1}having Relationship as MAX($p(S_n.\text{option}/S_{n-1}.\text{option})$) (or) MAX($p(S_1.\text{option})$)
8. TRAVERSE_DIG(S_{n+1}, S_n.option)
9. END IF
10. ELSE:
11. RECOMMEND USER the settings values stored in the leaf node S_n
12. END IF

The relationships are used to resolve uncertainties during selection of the most probable remaining settings. Assuming, S_n is the setting with higher Entropy value compared the setting S_{n+1}, where layer n comes first and $n+1$ comes next when traversing DIG. DIG will have a maximum of eight levels with each level representing one of the 8 WM settings. Depending on the entropy measure of the 8 settings in the DPT (Table 2), each WM DIG representation will have up to a maximum of 8 graph levels. The relationship R between nodes in S_n and S_{n+1} for DIG is shown below in (3), where the first value in the tuple is the setting option selected on the WM and second value is the ''path probability'' defined as $p(S_1.option)$ or $p(S_n.option/S_{n-1}.option)$ based on whether current node is level *1* or level n respectively. The exception is the root node in the DIG, which has no parent node and the second value is denoted by the probability of setting option $p(S_1.option)$. For example in Fig. 4, if 'Course' is setting S_1, then the two possible options denoted by $S_1.option$ are 'Super Speed' and 'Cotton'.

$$R\left(S_n, S_{n+1}\right) = \begin{cases} \left(S_n.option, p\left(\dfrac{S_n.option}{S_{n-1}.option}\right)\right), & if\ n \neq 1 \\ \left(S_1.option, p\left(S_1.option\right)\right), & if\ n = 1 \end{cases} \tag{3}$$

The example DIG representation (Fig. 4) generated from Table 2, shows that S_1 is ''Course'' and S_2 is ''Water Temperature''. The 'Course' setting has the highest entropy and is placed at the first level in the DIG. The settings ''Water Temperature'' and ''Spin Level'' have lower entropies and are placed at level 2 and 3 respectively. The remaining settings all have entropies of zero and hence they can be clubbed and placed in a single leaf node. By assigning these settings to the leaf node, the algorithm will ensure there are no interactions required for applying these settings. Each non-leaf node in the graph denotes one interaction to be made with the user. Based on the previous usage, the user needs to have 2 to 3 interactions to resolve any uncertainties. The 2-Tuple relationships (Super Speed, 0.4545) and (cotton, 0.5455) between S_1 and S_2 denote the uncertainty that needs to be resolved when moving from level 1 to level 2. The conditional probabilities in (3) are computed by using the '8-settings options' probabilities generated in Table 2. For example, the conditional probability between level 2 node and leaf node, $p(S_2.option = 60\ /S_1.option = \text{'SuperSpeed'})$ is computed as 0.1. During traversal of each level on the DIG starting from the root node, an interaction happens with the WM user to determine the path to reach the leaf node. We have explained how traversal works in the next section.

C. AUTO-COMPLETION USING DEVICE INTERACTION GRAPH TRAVERSAL

Each node on the DIG denotes an uncertainty that has to be resolved when the WM settings are being selected by the user. During each setting selection, the traversal of nodes on the DIG start from the root to the leaf node. As shown in Algorithm 2, the input parameters are the current DIG traversed state and next input selected by the user. Based on the current input selection by the user, a single path from the node is chosen and the other paths are disabled. The node selected by the user can be on any level on the DIG. When the user selects an option for the setting, the path with the maximum probability as shown in (3) is chosen and the remaining paths are disabled for that level. This ensures that there is only one path on the graph for that level, during future recursions on the same DIG nodes. The option on the path with maximum probability will be value applied as option to the setting. There will be up to a maximum of 8 recursions from the root to the leaves till the user's preferred settings are recommended. In the case of Fig. 6, there are 2 or 3 recursions of the TRAVERSE_DIG() function.

When the WM is switched on, the TRAVERSE_DIG() function is called. The most probable sequence on the DIG representation will be recommended after the multiple recursions of the function. Each time after the user selects the setting, the selected option and the current DIG state are passed as parameters to the function TRAVERSE_DIG() and

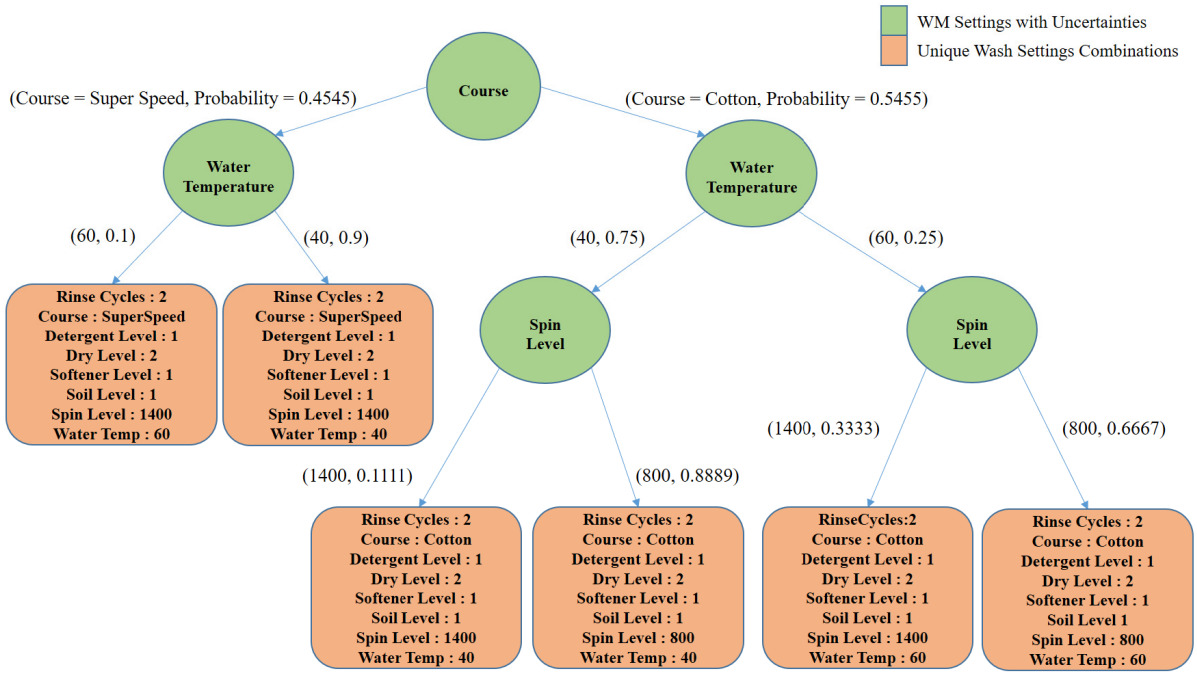

FIGURE 6. WM device interaction graph representation. This directed acyclic decision graph representation is generated for each WM individually. This representation will be used for settings auto-completion during user interactions.

then the function is executed. The algorithm will ensure that the most probable sequence which include the settings that are already selected are recommend to the user after each user selection.

In the case of auto-completion for voice interactions, the options for certain settings are mentioned in the voice command. The nodes for these settings are identified on the DIG. The TRAVERSE_DIG() enables only specific paths for these nodes based on the option mentioned by the user. For the remaining nodes on the DIG, where there are unresolved uncertainties, the paths with the highest probabilities are chosen and recommended as settings on the WM. This automatic recommendation ensures that there will be no further interactions with the user. The complete WM settings are enabled in a single step. In this case, the accuracy of the recommendation may not be 100%, as there are still unresolved uncertainties. But the DIG representation ensures that the options with the highest probabilities are applied for the settings.

V. EXPERIMENTS AND RESULTS

We have used the same data analyzed in Section III, for generating the results in this section. We have used real world data processed from 158,213 WM over a period of 6 months having over 10.5 million washes. The evaluation results shown in this section have used the first 5 months WM data from the 158,213 devices as the training data and 6^{th} month as the testing data. The DIG representation has been generated (using Algorithm 1) for each WM using 5 months of data. The 6^{th} month data has been used for testing the DIG traversal algorithm (Algorithm 2). The test data has been used to validate the accuracy of the DIG representation that was generated using the training data. By using the experiments and results in this section, we will show that the DIG representation is useful in recommending auto-completion settings for WM.

Fig. 7 shows the insight for average number of settings changed per WM (vs) the total WM count while applying the auto-completion algorithm. The results were generated by executing the DIG traversal algorithm on the test data. For a total of 47,114 WM, the average number of settings changed per WM was between 0 and 1. The value 0 here denotes the WM user has accepted the recommended settings without any further changes. Hence for a WM user who mostly accepted the first recommendation by the algorithm without any further changes of his own, the average settings changed per WM would be closer to 0. The other case would be a community WM or a WM in a large family home, where there would be a different user each time a WM was used. Each different user would have changed all the settings of the WM, every time WM had been used. Since there was no repetition pattern among the different user's using the WM, the algorithm accuracy for recommendation is very low in such a case. This can be observed in Fig. 7, where the average number of interactions are greater than 4. The average number of auto-completion interactions across all WM has come up to be 3.132.

Next, the accuracy statistics of applying the DIG traversal algorithm against all the WM is shown in Table 3. The evaluation criteria used for computing the statistics is: if the complete set of 8-setting options in the test data (selected by

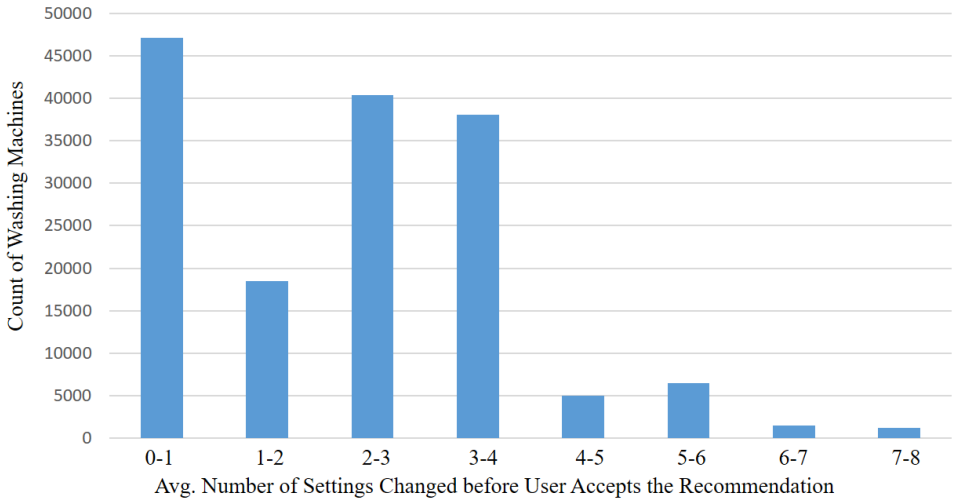

FIGURE 7. Average count of settings changed by the user (vs) count of WM, after applying DIG. For each setting that is changed there could be multiple options that could be selected by the user.

TABLE 3. Auto-completion algorithm recommendation accuracy metrics on test data.

Evaluation Criteria	Evaluation Metric
Total Number of WM	158,213
Avg. Recommendation Accuracy of all the WM	89.79%
Total Number of WM having accuracy > 90%	88,174
% of WM having Recommendation Accuracy >90%	55.73%
Total Number of WM having accuracy > 85%	116,859
% of WM having Recommendation Accuracy >85%	73.86%
Total Number of WM having accuracy > 80%	135,576
% of WM having Recommendation Accuracy >80%	85.69%

the user) is part of the DIG representation, then the result is a SUCCESS. If the 8 settings options selected by the user in the test data was not part of the DIG representation then the result was a FAILURE. A total of 88,174 WM or 55.73% of all WM had accuracy >90%, 116,859 WM or 73.86% of all WM had accuracy >85% and 135,576 WM or 85.69% of all WM had accuracy >80%. These results show that the most of the WM users had followed the same WM usage pattern that was provided in the DIG representation. The DIG representation was generated using 5 months of data and the average accuracy of auto-completion recommendations for the 6^{th} month usage was 89.79%. In cases where the WM user selected a new setting or option (which was previously unchanged), the recommendation would have not worked.

Next, we have considered existing algorithms using sequence pattern mining approaches that would recommend the complete 8-setting options for the WM. There are existing sequence pattern mining approaches that can mine and provide the most frequently used sequence patterns from the data [29]. We have considered the 8-settings options as a sequence of data and applied the Most Frequent Sequence Pattern Mining (MFSPM) solution on it. MFSPM had generated the Top-1 most frequently used sequence for the WM. The accuracy of Top-1 MFSPM was 36.84% as shown in Table 4. This means that the recommendation for the 8 settings was correct only 36.84% of the times. The remaining 73.16% times the user made at-least 1 change on top of the recommendation. The accuracy of Top-5 MFSPM had come up to 69.92%. This implied that even if we were to recommend the Top-5 usage sequences one after another, the average accuracy across all WM would have been 69.92%. Next we have considered a solution that linearly weights the time series WM data to give preference to the most recent data is described in [30]. In this paper, we have combined the approaches of linearly weighting time series data [30] with a frequent pattern mining approach from [29] to derive the "Frequent Linear Weighted Sequence Pattern Mining" solution (FLWSPM). For FLWSPM, the 8 settings for each wash are combined as a sequence. Distinct labels are assigned to the different wash settings sequences for each WM. The most recent wash label are assigned a higher weight compared to the older wash labels. For example, let us assume that there are 2 wash sequences WS1 and WS2, repeated by the user during the last 5 washes. The set of labels assigned for the 5 washes will be given by {WS2, WS2, WS1, WS1, WS1}. The label WS2 denotes the most recent wash and WS1 denotes the oldest wash in the time sequence. If we linearly weight the labels, based on the time sequence, then we get the set of weights as {5/15, 4/15, 3/15, 2/15, 1/15}. Finally, we get the probability of recommending WS2 and WS1 using FLWSPM as 0.6 and 0.4 respectively. For the same example above, if we use MFSPM, then we get the probability of recommending WS2 and WS1 as 0.4 and 0.6 respectively. This is because MFSPM weights the occurrences of labels equally. FLWSPM works better, when the user tends to repeat WM settings from most recent washes. By using FLWSPM we have evaluated whether the user repeated the previous wash with high probability or if the most recent washes had a higher impact in the current wash

TABLE 4. Comparison of proposed DIG with pattern mining algorithms.

Evaluation Criteria	Top-1 MFSPM	Top-5 MFSPM	FLWSPM	DIG Traversal	DIG Traversal	DIG Traversal
Total WM Count	158,213	158,213	158,213	158,213	158,213	158,213
Avg. No. of Interactions for All WM	0	0	0	1	2	3.132
% Accuracy for Avg. No. Interactions for all WM	36.84%	69.92%	43.10%	52.87%	75.51%	89.79%

TABLE 5. Comparison of proposed DIG with ML and DL approaches.

Evaluation Criteria	Random Forrest	FCNN	LSTM	DIG Traversal
Approach	Machine Learning	Deep Learning	Deep Learning	Decision Graph
Total WM Count	158,213	158,213	158,213	158,213
% Accuracy for all WM	52.08%	52.84%	52.12%	89.79%
Avg. No. of Interactions	0	0	0	3.132

settings selection. The most recent and frequently repeated 8-setting options used were given higher weightage than the less recent and less frequently used settings. The average accuracy of FLWSPM came up as to be 43.10% as shown in Table 4. Both Top-1 MFSPM and FLWSPM approaches required zero inputs from the WM user. Since FLWSPM had given higher average accuracy than Top-1 MFSPM, we can consider the default recommendation on the WM as the output of FLWSPM algorithm. If the WM user does not prefer the default settings and options recommended, then the user will start changing the settings manually. The goal of the DIG algorithm is to ensure that the users will have to do minimal interactions before reaching their preferred settings options. After the first change was made by the user and the DIG traversal algorithm was applied to recommend the remaining 7-settings options, the average accuracy improved to 52.87% as shown in Table 4. After the user made 2 changes and the DIG traversal algorithm was executed to recommend options for the remaining 6 settings, the average accuracy improved to 75.51%. Overall across all the WM being considered, if the user made an average 3.132 changes on the settings, the overall accuracy reached a maximum of 89.79% for the proposed DIG traversal algorithm.

Next, we have considered ML and Deep DL approaches to recommend the 8-settings for each WM. For predicting the preferred wash settings using ML and DL model based approaches, we have considered Random Forrest [34], Fully Connected Neural Network (FCNN) [27], [28], [35] and LSTM [36]. All these models have been generated using the same input data available in WM DST. The comparative results have been captured in Table 5. For the Random Forrest and FCNN models input data, all consecutive washes (in WM DST) from all WM have been segmented into windows of N washes. The 8 settings used for the first *N-1* washes have been used as input features and the 8 settings of the N^{th} wash has been used as the Ground Truth for the prediction target. There are a total of *(8*N)* input features and 8 output targets. For the LSTM model, the 8 settings for each wash are combined as a sequence. Distinct labels are assigned to the top recurring sequence to the least recurring sequence. For example, if there are a total of 10 distinct settings sequences are identified for a WM, then each sequence is labeled from *S1*, *S2* ... *S10*. *S1* is the top recurring sequence in the DST for the WM. *S10* is the least recurring sequence in the DST. LSTM model has been trained by feeding the *N-1* sequence labels assigned for all distinct setting sequences from the WM historical washes. FCNN model generated learns weights from input features to predict target output class. A Random Forest approach builds a collection of decision Trees using the input data. LSTM (Long Short Term Memory) network is a kind of Recurrent Neural Network (RNN) capable of identifying long term dependencies in sequential time series wash data. For all models, previous wash window sizes of 5, 10, 15 and 20 have been considered and all the window sizes have given almost the same result. Therefore, we chose last 5 washes (to reduce model complexity), as the input window size for DL modeling. Among the DL models, all models had shown similar prediction accuracies, with the best accuracy of 52.84% observed for the FCNN solution. This is substantially lower compared to the 89.79% accuracy obtained for DIG traversal. The ML and DL approaches recommend the most probable prediction without considering any interactions. The DIG considers the partial settings that are provided by the user during interactions, to recommend with higher accuracies.

The Pattern Mining, ML and DL approaches that we have considered so far, recommended the most probable 8 settings for each WM. From the results, we have observed that recommending the 8 settings together had resulted in lower accuracies. The computational complexity and model memory consumption of ML and DL approaches for recommending the 8 settings are also high. Such computational complexity and memory may not be supported on most WM hardware. For the proposed solution, the only complexity during recommendation lies in the DIG traversal logic. The DIG representation achieved high accuracy by taking the inputs that were already provided by the user during voice/manual interactions with the WM. This helped in achieving higher accuracy compared to the existing approaches.

VI. CONCLUSION

We have discussed a Device Interaction Graph representation using a directed acyclic decision graph for storing the past user-device interactions knowledge. Human tendency is to use a certain combination of settings together when using devices, and DIG representation captures this combination relationship. We have also explained the DIG traversal algorithm that uses the DIG representation to enable auto-completion of the settings in real time during the user interactions. The DIG traversal algorithm provided an accuracy of 89.79% in recommending WM settings through the auto-completion algorithm with an average of 3.132 user interactions across all WM. Using the DIG representation on testing data, we have also shown that the auto-completion recommendations have been accepted over 90% of the times by 55.73% of WM users and 80% of the times by 85.69% of the users. With this proposed solution, we have shown that knowledge generated from device data can be used to personalize and simplify the device usage experience. During voice interactions, we have shown that the most probable recommendations are applied in a single step after the first interaction. For future research, the proposed solution can be customized and applied to other IoT enabled devices such as AC, Microwave oven, Industrial machines etc. Another area of research would be in the direction of lightweight interactive ML and DL solutions which takes in partial inputs (that were already provided by the user) to improve the prediction accuracies. The solution can be adopted and used in IoT automation scenarios, where devices/machines have multiple settings which the user has to select before enabling operations. Auto-completion of settings for enabling device operations, during voice interactions, has important benefits in the Conversational AI domain.

REFERENCES

[1] W. A. Jabbar, T. K. Kian, R. M. Ramli, S. N. Zubir, N. S. M. Zamrizaman, M. Balfaqih, V. Shepelev, and S. Alharbi, "Design and fabrication of smart home with Internet of Things enabled automation system," *IEEE Access*, vol. 7, pp. 144059–144074, 2019, doi: 10.1109/ACCESS.2019.2942846.

[2] W. T. Hartman, A. Hansen, E. Vasquez, S. El-Tawab, and K. Altaii, "Energy monitoring and control using Internet of Things (IoT) system," in *Proc. Syst. Inf. Eng. Design Symp. (SIEDS)*, Charlottesville, VA, USA, 2018, pp. 13–18.

[3] C. Kaiwen, A. Kumar, N. Xavier, and S. K. Panda, "An intelligent home appliance control-based on WSN for smart buildings," in *Proc. IEEE Int. Conf. Sustain. Energy Technol. (ICSET)*, Hanoi, Vietnam, Nov. 2016, pp. 282–287.

[4] D. Wu, "Research on the micro-interactive interface design of intelligent washing machines in IOT environment," in *Proc. 22nd Int. Conf. Autom. Comput. (ICAC)*, Colchester, U.K., Sep. 2016, pp. 479–487, doi: 10.1109/IConAC.2016.7604966.

[5] V. Kepuska and G. Bohouta, "Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, Jan. 2018, pp. 99–103, doi: 10.1109/CCWC.2018.8301638.

[6] W. Ai-Zhen and R. Guo-Feng, "The design of neural network fuzzy controller in washing machine," in *Proc. Int. Conf. Comput., Meas., Control Sensor Netw.*, Taiyuan, China, Jul. 2012, pp. 136–139, doi: 10.1109/CMCSN.2012.35.

[7] J. Gu and D. Qiang, "The design of intelligent washing machine controller based on FPGA," in *Proc. 5th Int. Conf. Instrum. Meas., Comput., Commun. Control (IMCCC)*, Qinhuangdao, China, Sep. 2015, pp. 1529–1532, doi: 10.1109/IMCCC.2015.324.

[8] G.-J. Wang, Z.-X. Wang, H.-D. Xu, and X.-G. Zhang, "The application of fuzzy-neural network based on HGA in the washing machine control system," in *Proc. Int. Conf. Intell. Comput. Technol. Autom.*, Changsha, China, May 2010, pp. 702–705, doi: 10.1109/ICICTA.2010.49.

[9] G. A. Susto, G. Zambonin, F. Altinier, E. Pesavento, and A. Beghi, "A soft sensing approach for clothes load estimation in consumer washing machines," in *Proc. IEEE Conf. Control Technol. Appl. (CCTA)*, Copenhagen, Denmark, Aug. 2018, pp. 1252–1257, doi: 10.1109/CCTA.2018.8511398.

[10] G. Zambonin, F. Altinier, L. Corso, M. Sessolo, A. Beghi, and G. A. Susto, "Soft sensors for estimating laundry weight in household heat pump tumble dryers," in *Proc. IEEE 14th Int. Conf. Autom. Sci. Eng. (CASE)*, Munich, Germany, Aug. 2018, pp. 774–779.

[11] S. Kaler and R. Gupta, "The design of intelligent washing machine controller based on FIS & ANFIS," in *Proc. Int. Conf. Inf., Commun., Instrum. Control (ICICIC)*, Indore, India, Aug. 2017, pp. 1–6, doi: 10.1109/ICOMICON.2017.8279163.

[12] M. Maggipinto, E. Pesavento, F. Altinier, G. Zambonin, A. Beghi, and G. A. Susto, "Laundry fabric classification in vertical axis washing machines using data-driven soft sensors," *Energies*, vol. 12, no. 21, p. 4080, Oct. 2019, doi: 10.3390/en12214080.

[13] G. Zambonin, F. Altinier, A. Beghi, L. S. Coelho, N. Fiorella, T. Girotto, M. Rampazzo, G. Reynoso-Meza, and G. A. Susto, "Machine learning-based soft sensors for the estimation of laundry moisture content in household dryer appliances," *Energies*, vol. 12, p. 3843, Jan. 2019.

[14] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.

[15] C. W. Tsai, C. F. Lai, M. C. Chiang, and L. T. Yang, "Data mining for Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 77–97, 1st Quart., 2013.

[16] D. Patnaik, P. Butler, N. Ramakrishnan, L. Parida, B. J. Keller, and D. A. Hanauer, "Experiences with mining temporal event sequences from electronic medical records: Initial successes and some challenges," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 360–368.

[17] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining Knowl. Discovery*, vol. 1, no. 3, pp. 259–289, 1997.

[18] W. Zhou, H. Liu, and H. Cheng, "Mining closed episodes from event sequences efficiently," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Germany: Springer, 2010, pp. 310–318.

[19] J. Han, M. Kanber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufman, 2012.

[20] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[21] H. Wang, S. Lu, C. Zhang, Q. Wang, and F. Xu, "Timing-IdeaGraph: A directed cognition graph approach for decision making based on temporal event sequences," in *Proc. IEEE 20th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, Nanchang, China, May 2016, pp. 322–326.

[22] R. D. L. Briandais, "File searching using variable length keys," presented at the Western Joint Comput. Conf., Mar. 1959.

[23] T. Gueniche, P. Fournier-Viger, and V. S. Tseng, "Compact prediction tree: A lossless model for accurate sequence prediction," in *Proc. Int. Conf. Adv. Data Mining Appl.* Berlin, Germany: Springer, 2013, pp. 177–188.

[24] C. Simon, J. Meessen, D. Tzovaras, and C. De Vleeschouwer, "Using decision trees for knowledge-assisted topologically structured data analysis," in *Proc. 8th Int. Workshop Image Anal. Multimedia Interact. Services (WIAMIS)*, Jun. 2007, p. 2.

[25] S. Tsang, B. Kao, K. Y. Yip, W.-S. Ho, and S. D. Lee, "Decision trees for uncertain data," in *Proc. IEEE 25th Int. Conf. Data Eng.*, Mar. 2009, pp. 441–444. 10.1109/ICDE.2009.26.

[26] C. Liang, Y. Zhang, and Q. Song, "Decision tree for dynamic and uncertain data streams," in *Proc. 2nd Asian Conf. Mach. Learn. (PMLR)*, vol. 13, 2010, pp. 209–224.

[27] R. Kannan and M. S. Roy, "AC cooling time prediction using common representation model," *IEEE Access*, vol. 8, pp. 131534–131544, 2020, doi: 10.1109/ACCESS.2020.3009467.

[28] R. Kannan, M. S. Roy, and S. H. Pathuri, "Artificial intelligence based air conditioner energy saving using a novel preference map," *IEEE Access*, vol. 8, pp. 206622–206637, 2020, doi: 10.1109/ACCESS.2020.3037970.
[29] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Sci. Pattern Recognit.*, vol. 1, no. 1, pp. 54–77, 2017.
[30] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*, vol. 104. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
[31] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, p. 67, 2016.
[32] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.
[33] X. Xie, D. Wu, S. Liu, and R. Li, "IoT data analytics using deep learning," 2017, *arXiv:1708.03854*. [Online]. Available: http://arxiv.org/abs/1708.03854
[34] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[35] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of IoT networks using artificial neural network intrusion detection system," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, 2016, pp. 1–6, doi: 10.1109/ISNCC.2016.7746067.
[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

RAMASAMY KANNAN (Senior Member, IEEE) received the B.E. degree in electronics and communication engineering from the University of Madras, India, in 2002, and the master's degree in digital communications from the University of Kiel, Germany, in 2004. He is currently an AI Architect with the Samsung Research and Development Institute India, Bengaluru. Various algorithms designed and developed by him are now commercialized in different Samsung products. His research interests include AI for digital appliances, vision intelligence, signal processing, image processing, multi-modal interactions, and sensor algorithms. He has received the Merit Award in Samsung Best Paper Award Contest, in 2014, and the Bronze Award in Samsung Best Paper Award Contest, in 2015.

SAI HARISH PATHURI received the B.Tech. degree in computer science and engineering from the Manipal Institute of Technology, Manipal, India, in 2019. He is currently a Research Engineer with the Samsung Research and Development Institute India, Bengaluru. His research interests include cloud services, big data analytics, and the IoT intelligence.

DEEPAK KUMAR received the B.Tech. degree in information technology from the Cochin University of Science and Technology, Kochi, India, in 2011. He is currently a Chief Engineer with the Samsung Research and Development Institute India, Bengaluru. His research interests include natural language generation, conversational AI, vision intelligence, and distributed computing.