Recommendations for Creating Trigger-Action Rules in a Block-based Environment

Andrea Mattioli
Human Interfaces in Information Systems Laboratory
CNR-ISTI
Pisa, Italy
andrea.mattioli@isti.cnr.it

Fabio Paternò
Human Interfaces in Information Systems Laboratory
CNR-ISTI
Pisa, Italy
fabio.paterno@isti.cnr.it

ABSTRACT

Given the growing adoption of IoT technologies, several approaches have been presented to enable people to increase their control over their smart devices and provide relevant support. Recommendation systems have been proposed in many domains, but have received limited attention in the area of End-User Development (EUD). We propose a novel approach for formulating recommendations in this area, based on deconstructing trigger-action rules into sequences of elements and the links between them. For this purpose, we propose a solution inspired by methods aimed at addressing the sequence-prediction problem. We have used this approach to provide users with two different types of recommendations: full rules for the one being edited, and parts of rules relevant for the next step to take in order to complete the current rule editing. In this paper, we present the design and a first evaluation of the two different possibilities to generate and display recommendations in a block-based EUD environment for creating automations for Internet of Things (IoT) contexts.

KEYWORDS

End user development, Recommendation systems, Recommendations for personalization, Internet of Things, Trigger-Action programming

1 Introduction

The Internet of Things is a very pervasive technological trend. It is estimated that there are 30 billion connected objects and devices in 2020. This is an area that benefits in particular from the adoptions of solutions that empower users in controlling the automations involving all such objects. In this perspective, it is important to provide users with editors that allow them to define the desired automations for their daily environments. Trigger-Action Programming (TAP) is an EUD approach that has been considered as an effective approach for this purpose since it does not require any particular algorithmic ability, and allows users to easily indicate when a rule should be triggered and what the associated effect should be. It is based on the rule metaphor and takes the form of conditional statements, like "if *something happens*, then *activate some behaviour*". IoT devices and Web services can be used both in the trigger (the "if" part) and in the action (the "then" part) of the rule. Previous studies (Ur et al., 2016; Cabitza et al., 2017) found that a rule-based approach is easily understandable, and people without programming experience can create their programs which can contain multiple triggers and actions.

Tailoring environments are the instruments used to allow users to indicate how to connect their devices and services. Various visual tools in both commercial (e.g. IFTTT[1], Zapier[2]) and research, e.g. TAREME (Manca et al, 2019), EFESTO (Desolda et al., 2017) contexts have been put forward in this area. The goal is to facilitate the end-user creation of trigger-action rules that determine when and how the automations should be performed. However, this end user development process still suffers from some issues, also because of the vast and increasing number of IoT devices, and possible associated personalization rules. Another problem derives from the incorrect transmission of the timing aspects of the rule elements. As examined in (Huang and Cakmak, 2015), with "trigger" we can refer to an event that happens in a punctual moment in time (when user enters a room, when it starts to rain, when kitchen temperature exceeds 30 degrees, at 8 o'clock), or to a condition or state that lasts for a longer period of time (while user is inside a room, as long as it's raining, until kitchen temperature is over 30 degrees, between 20:00 and 23:30). Similar timing aspects can be found in actions, that can be almost immediate such as send a message, can lasts for a period and then terminate, or can change the state of a device until some other agent act on that device. Simpler rule editors (such as IFTTT) do not make a distinction on these types, but more advanced editors have to express it clearly in order to prevent users' errors in their use. The timing aspects of rule elements may lead to unexpected system behaviours and bugs (Brackenbury et al., 2019), like the different interpretations that users can have of the combination of a condition type trigger with a sustained action (Huang and Cakmak, 2015). Further

[1] https://ifttt.com/home
[2] https://www.zapier.com

challenges emerge in scaling up the TAP approach to a practical setting, where more device and services are used, and multiple rules can interact with each other.

In this shift from "consumer cultures" towards "cultures of participation" (Fischer, 2011) where people have access to the means to solve meaningful problems, different attempts have been made to support users during the rule creation process. Some of them are the introduction of metaphors and visual clues (Danado and Paternò, 2012, Desolda, Ardito and Matera, 2017, Corno, De Russis and Roffarello, 2019b), methods to specify the timing relationships of rules (Barricelli and Valtolina, 2017; Zhang et al., 2019), the implementation of debuggers and simulators (Manca et al., 2019, Nacci et al., 2018, Corno, De Russis and Roffarello, 2019a), and the use of natural language (Coutaz and Crowley, 2016; Corno, de Russis and Roffarello, 2020). We believe that the introduction of some recommendation support can be an effective method to help users to find the proper elements to model the desired behaviour. Recommendations can provide valid suggestions for both beginners and more advanced users, in that the former can benefit from being guided to discover the tailoring environment being used by seeing the structure of the rules and the possible next steps to take in order to complete the editing process, while the latter can discover new functionalities on their own. Suggestions can be also useful to support user's reasoning about temporal aspect, providing examples that guide them toward the assimilation of the correct mental model.

Recommendation systems (RS) have the purpose of determining the chance that a user will like an item. This can be used to propose items of interest for that user, and to reduce the cognitive load needed for the selection (Ricci et al., 2011). The typical approach to such problems is to consider only one user-item interaction (Quadrana, Cremonesi and Jannach, 2018), such as ratings, and aims to predict the score for the unknown items. However, different application contexts require different approaches. For example, when the object argument of the recommendation is not unitary but comprised of multiple parts, or when a user can interact with an item multiple times. These situations are relevant to recommendations for the TAP context because rule editing is a process comprised of multiple steps, where users should receive an appropriate suggestion depending on the current editing stage, and considering the rule elements already inserted. In this perspective, an RS for this context can be considered a type of sequence-aware RS, where we know the previous interactions of the user with the system (i.e. in our case in the form of saved rules). Another aspect to consider is the richness of the data available in a scenario. This may vary from the sequence of interactions alone, to the sequence enriched by additional information, such as the type of the sequence element or the items which it refers to. These data can be exploited to provide more targeted suggestions. Hence, we propose a solution for recommendations in TAP context based on sequence prediction models that also integrates the feature of the single rule elements.

The remainder of the paper is organized as follows: we start with a review of work relevant for this research. Next, we describe some background information regarding the visual editor that we have considered as starting point for our research and presents the design choices adopted. Section 4 describes the two implemented methods to obtain and present recommendations in a trigger-action system, while Section 5 reports on a first user evaluation of the proposed environment. Section 6 discusses the results obtained through the test, and their implications. Lastly, we draw some conclusions from the work carried out and provide indications for future work.

2 Related Work

As discussed in (Corno, De Russis and Roffarello, 2019b), three visual paradigms stand out as the most relevant for EUD in IoT scenarios: guided approaches (based on Wizard-like support), block-based, and dataflow. Some studies (Rough and Quigley, 2017; Weintrop, 2019; Weintrop et al., 2018) found block-based programming suited for education and domains whose concepts can be clearly subdivided in elements and expressed as blocks. Environments based on blocks are used as an introduction to programming (Scratch, Snap), as alternative interfaces for Arduino (Ardublock, S4A - Scratch 4 Arduino, BlocklyDuino), and for developing mobile applications (MIT App Inventor).
A contribution based on the puzzle metaphor is (Danado and Paternò, 2012), where jigsaw pieces are associates with predefined command sequences, and used to create mobile applications. In (Guilly et al., 2015) a block-based system is adopted to define scenarios, and associated constraints, as abstractions over the underlying ECA language. The creation occurs via drag-and-drop of device icons into one of the three empty spaces, to instantiate a new event, condition, or action. In (Rough and Quigley, 2017), blocks are used to allow researchers to develop custom applications for experience sampling. The visual affordance of blocks is used to support creation of rules with one event trigger and an optional number of state conditions, in a similar fashion to ECA rules. An explicit ECA structure has the advantage of disambiguating at the interface level the distinction between event and state triggers (Rough and Quigley, 2017). SmartBlock (Nayeon, Chang, and Choi, 2018) is a block-based interface for structuring rules for home automation in the event-condition-action (ECA) format. A further contribution that rely on the puzzle metaphor is (Corno, De Russis and Roffarello, 2019b), where a tool to compose and debug trigger-action rules was introduced. The tool can in some cases identify bugs in real time during rule editing, provide users with visual feedback on the problematic elements, and some explanations about how to solve it. Relevant works that adopted a more guided approach to allow for the use of multiple triggers are (Ghiani et al., 2017) and (Desolda, Ardito and Matera, 2017), the former providing natural language feedback of the rules created, the latter uses an approach based on the 5W model and on a graphical grouping system together with logical forms, such as Disjunctive and

Conjunctive Normal Form. Other block-based approaches can be found in different but related fields of study. In the visual programming-based toolset Quando (Stratton, Bates, and Dearden, 2017), a trigger-action approach is used to allow cultural heritage professionals to enhance visitor exhibits with digital interactivity, combining blocks from a library to define more complex behaviours. Coblox (Weintrop et al., 2018) is a block-based environment proposed to allow users to specify procedures for common industrial robot operations.

Regarding the use of RS in EUD systems, they can be used to help users to make well-formed decisions in the rule composition process, in particular when the number of possible items to recommend is large, or the contextual information available is too limited (Haines et al. 2010). Some solutions have been initially proposed in EUD approaches for Web mashup contexts, such as (Carsten et al., 2012; Chowdhury et al., 2013). More recently, in (Srinivasan et al., 2016) the authors put forward an approach to identify the most relevant set of rules for users of a system like IFTTT, introducing a top-k RS which suggests a set of items obtained by mining the user's habits. From the smartphone usage logs they derived some sets of preconditions that describe the state before the activation of a rule, analyzing the periods of time in which contexts and actions frequently occurs together. An approach based on semantics has been proposed in (Tomlein et al, 2017), where a reasoner has been applied to rule elements to identify if an installation is capable of executing that rule. Afterwards, possible rules suggestions are ranked based on the similarity of sensor data and installations. RecRules (Corno, de Russis and Roffarello, 2019c) adopted a semantic graph approach with the purpose of abstracting from the technological details of devices (brands, manufactures), to suggest rules based on the behaviour that the user wants to achieve. The model considers user's implicit and explicit feedback toward a rule, and matches the user interests with the rule content using different path-based features (collaborative information, technology-based similarities, and functionality-based similarities). HeyTap (Corno, de Russis and Roffarello, 2020) proposes a conversational agent to recommend rules based on the user preferences and intent. It operates over the EUPont model, which provides a semantic representation of IoT concepts useful for EUD, such as functionalities of triggers and actions, contextual information, user preferences. The user input is used along with the contextual information about the entities to infer and suggest a set of trigger-action rules.

Related contributions oriented toward the IoT context are (Wang et al., 2018), where a general method to recommend rules which considers user-item-device interactions is introduced. Rules has been divided in categories, i.e. "single trigger, single action", "single trigger, multiple actions", "multiple triggers, single action" and "multiple triggers, multiple actions". Also, devices have been categorized based on their functionality. Basic assumptions are that users who install similar devices share similar preferences, and rules which require similar types of device may share similar functions. Recommendations are generated applying a collective matrix factorization scheme on the user-rule-device information. (Jeong et al. 2019) propose a framework to analyse the device-usage logs of smart devices in the IoT context. Segmentation is also applied to generate recommendations, integrating the individual user usage patterns with the ones of people who live in similar contexts. Our approach differs from the presented ones because we have conceived recommendation as a way to support users while creating a rule through a system that completes the rule under construction by providing suggestions about relevant triggers, operators and actions. Also, our focus was not only on finding a different and more suitable approach to generate suggestions, but also on investigating the most effective presentation modalities for such recommendations.

Applications of RSs based on sequence of actions can be found in different fields. One is personalized online learning, where (Intayoad, Kamyod and Temdee, 2020) applied a reinforcement learning-based approach on the sequence of learning objects consumed by students. Due to the complexity of the problem, instead of the full reinforcement learning problem they applied a contextual-bandit approach based on epsilon-greedy, which is suitable for a dynamic environment with a high grade of uncertainty. A solution for giving support in the cooking domain is proposed in (Nouri et al., 2020). The approach is a multi-step RS, that can make use of both the previous and next steps in the recipe to make decisions about the recommendation (if recommendation is needed, which type of additional information to present, in which format, where to deploy them). An approach for e-commerce is proposed in (Twardowski, 2016). In this setting, data available for recommend items are often limited to the sequence of user activities within that session. To obtain a next item recommendation, matrix factorization and Recurrent Neural Network (RNN) were used, where the latter can directly model the user sequential behaviour, and combine the user profile with short term, session-based intents. (Tavakol and Brefeld, 2014) used an approach based on factored Markov decision process (fMDP) to detect the user goal for a session, in a context where user feedback is implicitly evinced (e.g. from user clicks). In (Quadrana et al., 2017) a general approach based on RNN was calibrated to cope both with session-aware and session-based recommendations. In the context of our work, we found a hybrid approach based on different models more suitable; one to take in account the sequence of elements, one for their features.

3 The Design of the Proposed Solution

In the design and development of a novel solution for a block-based environment for trigger action-rules, we started from a rule editor of this type previously developed (Mattioli and Paternò, 2020). Block Rule Composer is a web-based tailoring environment whose purpose is to allow users to easily create rules specifying compound triggers and actions, using a block-based language implemented with the client-side

JavaScript Blockly[3] library. The focus of the environment is the end-user composition of rules using an approach which involves less constraints with respect to more guided wizard-based systems. The editor aims to facilitate the creation of rules by making explicit some important concepts of trigger-action rule editing. For example, the distinction between event and state type triggers is shown with a modal window each time a user selects a trigger type block, and the distinction between different action types (immediate, extended, sustained) is indicated via a tooltip.
The process of editing a rule consists of the selection of the elements of interest from a toolbox, which contains a hierarchical organization of the sensors, actuators, and Web services available. The composition occurs in a main workspace, where the selected elements can be dragged and dropped inside a pre-existing rule block. The "and", "or" and "not" operators can be applied to triggers, while actions can be joined using a "sequential" or "parallel" operator.

The prototype of this environment included a preliminary version of an RS based on the deconstruction of the trigger part of the rules into its constituent elements. At recommendation time, a network of the items related to the current trigger was generated and automatically analysed to find a suitable trigger rule part. This trigger part was then joined to an action part obtained using a collaborative/content-based approach, and the resulting full rule suggested. The introduction of the RS was well received during a first user test (with 12 people, without previous experience with personalization systems and programming) in terms of the relevance of the suggested rules (Mattioli and Paternò, 2020). However, several participants indicated that they would appreciate a more guided step-by-step recommendation system. Moreover, this preliminary solution would have caused scalability problems, as the rules in the dataset increased. From an analysis of previous work, a sequence-based RS appears to be a more suitable, yet unexplored solution for a recommendation system for personalization rules in IoT.

However, it should be considered that an RS solely based on presenting suggestions for the next step could suffer from some shortcomings:

- It may lose some of its usefulness with more experienced users, or when novice users have acquired more expertise with the rule editing process.
- An approach that also presents full rules can display within a single suggestion more rule elements and possibilities to combine them, giving a broader and possibly more serendipitous possibilities about how to compose rules.
- Providing only a rule element may be less helpful if it is not supported by its values. Instead, a full rule recommendation includes by default the values of its elements.

For these reasons, we designed an RS that, starting with a common sequence prediction algorithm, can provide suggestions with these features:

- Allow for providing both step-by-step and full-rule recommendations, with the possibility to easily switch between them, without the need to start the editing process over.
- Allow for obtaining a preliminary set of suggestions when a rule element has just been selected, and a refined one when the rule element under editing has been completed (e.g. specifying the next operator in the rule, or if the negation will be applied).

4 Recommending rules

4.1 Presentation

Starting from the environment cited in the previous section, we modified its presentation and underlying functionalities to make it more suitable to provide recommendations and support the users during the rule editing. The major additions to the user interface are the adaptive "next step" and the "suggestion type" bar. The new solution proposed in this paper is laid out in Figure 1.

[3] https://developers.google.com/blockly/

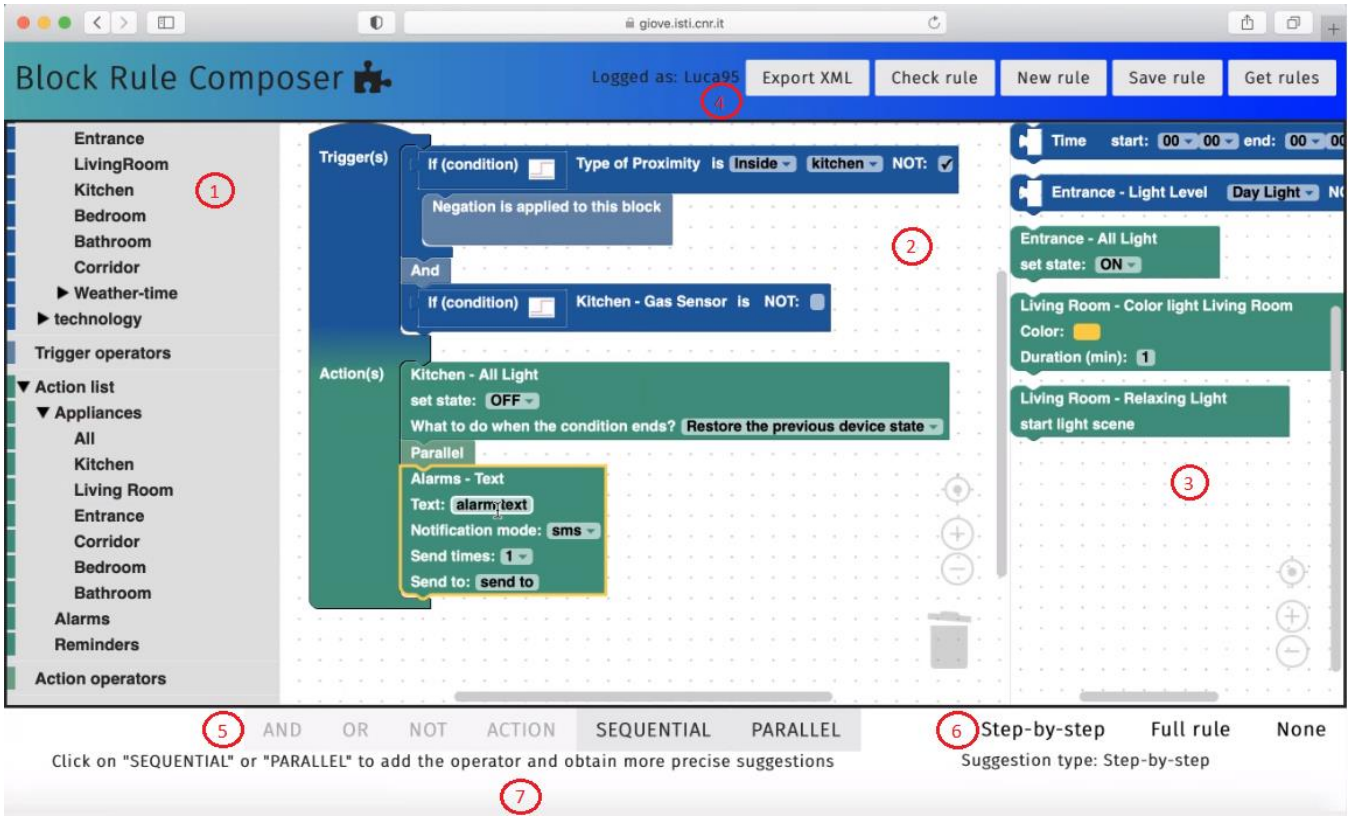

Figure 1: The main interface of the application with a rule under composition.

The main user interface of the editor consists of a toolbox (Fig. 1, circle 1), which contains the triggers, actions and operators that can be used; the main workspace (circle 2) where the arranging of blocks to compose a rule occurs; a secondary workspace (circle 3) where the RS suggestions are displayed; a "utility" toolbar (circle 4) that contains a list of buttons for the main operations possible on the editor (e.g. save and load rules); a "next element" toolbar (circle 5) to dynamically show the currently available operators , which can be selected to add them to the rule and activate the suggestions refined accordingly; a "suggestion type" toolbar (, circle 6) to select between step-by-step, full rule and no suggestions, and an "info and help" text area (circle 7) that shows information about a block and warnings if syntactic errors are detected.

The main improvements from the previous editor regards the methods to obtain and present recommendations. In the proposed solution, recommendations are generated considering the inputs provided by the user during rule composition. The inserted elements are used as input to a tree model, which has been trained with all the rules extracted from the available rules dataset. The obtained prediction consists of a list of viable next elements for the input sequence. Further techniques are then used to obtain full rules from these sequence elements (in the "full-rule" suggestions), or to further refine them (in "step-by-step" mode). Recommendations are then presented using the secondary workspace (see Figure 1). The user can switch between one mode and the other using a "recommendation type bar" in the lower part of the main user interface. There are five user-generated events that cause the activation of a recommendation: select a rule element from the toolbox, delete a rule element from the main workspace, add a rule element from the secondary workspace, select an element on the "next element" bar, and select an element on the "recommendation type" bar.

Step-by-step suggestion is the default method to present recommendations to the user. The generation of a next step suggestion involves obtaining from rules already saved in the repository a list of suggestions suitable for addition to the rule fragment currently under editing. This is achieved by acting on two different steps in the rule editing process. The first is to recommend an object as soon as the user selects a rule element from the toolbox and places it into the workspace. Otherwise, if an option from the "next element" bar has been selected, a more refined suggestion considering this further selection is generated.

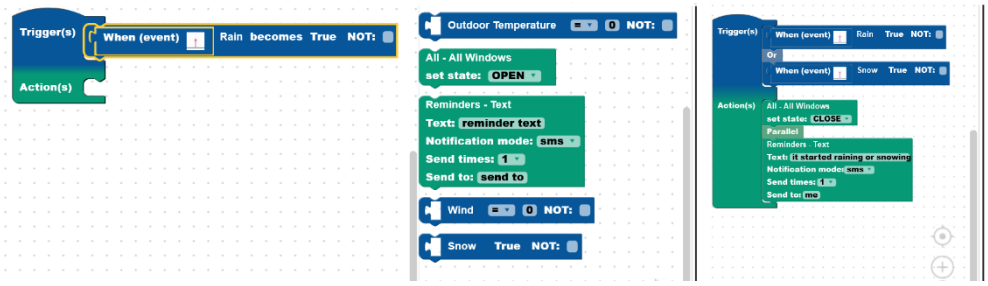

Figure 2: A comparison of the suggestions obtained when inserting "rain" into the workspace: in the middle the list of step-by-step suggestions, in the right one of the "full rule" recommendations.

Full rule suggestions are instead only generated when a rule element is inserted. Figure 2 shows examples of recommendations in both modes.

In both recommendation modes, the selection of a rule element from the suggestion workspace generates a copy of that block, which is then moved into the main workspace. If a suitable connection is available, this block is automatically placed on the correct slot in the main rule in the main workspace.

4.2 Underlying models

The main model adopted to obtain predictions for the next sequence element is the Compact Prediction Tree (CPT) (Gueniche, Fournier-Viger and Tseng, 2013; Guerniche et al., 2015). CPT is a lossless sequence prediction model, which uses all the elements of the sequences and can obtain predictions on unknown sequences. We chose to use CPT[4] because it has been tested on multiple datasets and obtained good and consistent accuracy performance. It is also fast to train, scalable, its operations can be represented using a tree structure, which can be a valuable feature in the explanation of recommendations. Furthermore, this algorithm is not a black box, so it can be easily tuned and extended. For this reason, it has been used together with other techniques to provide more suitable results for both recommendation modes. To use this model, the rules have to be deconstructed into a list of transitions. For example, if we compose this rule:

```
When (event) position becomes outside kitchen and if (condition) fridge door is open, close the fridge door and sequentially send a reminder via voice.
```

A sequence of transitions will be saved in the database used by the CPT model:

```
position => fridge door status => fridge door action => reminder
```

The technique used to refine the CPT results is multilayer perceptron, often simply called neural network (Ahmed et al., 2010), a model that rely on a stack of layers and a linear feeding between them. It has been successfully used in many classification and regression tasks. This model can be naturally adapted to our problem and, even if simpler than other approaches, can provide an enhancement on the results obtained by CPT by looking at all the features of the elements. Also, explanation techniques such as a partial dependence plot can be applied to represent the weight of each feature in the generation of the suggestions (Molnar, 2020). To adopt this technique, we used the ml5.js[5] library, a high-level interface to TensowFlow.js[6] On the specific, we used the neural network ml5 model, that relies on the TensorFlow sequential model. Before being used to train the model, data have been processed as an "element-attribute" list (see table 1). For each rule element, a row is inserted in the dataset, including each attribute of that rule's elements. The next operator and the next element are considered attributes of the current element. Data from this second dataset has been used as the training set for the neural network classifier. Categorical features have been transformed in vectors using one-hot encoding. At recommendation time, we obtain from the inserted rule element all the attributes of an item, except the "next element": in this way the problem is reformulated as a multiclass classification problem.

[4] We used the JavaScript implementation that can be found at https://github.com/ashubham/CPT
[5] https://ml5js.org/
[6] https://www.tensorflow.org/

Element	Next element	Type	Trigger type	Action type	Negation	Link
Position	fridge door status	Trigger	event	none	none	and
fridge door status	fridge door action	Trigger	condition	none	none	rule
fridge door action	reminder	Action	none	extended	n/a	sequential
reminder	none	Action	none	immediate	n/a	none

Table1: data about a single rule as used to train the neural network

4.3 Step-by-step suggestions

Step-by-step recommendations come in two different types, "standard" and "refined". The former are obtained via directly applying CPT on the sequence of rules elements inserted by the user and provide the top 5 results in the suggestion window. If no result is found, only the last inserted rule element is used to generate the recommendation. The latter type of recommendations occurs when the user has edited the rule element, so more information is available. These additional data are the type of trigger (event or condition), any possible negation applied to the trigger, the type of link that connects a rule element with the following one (and, or, rule, parallel, sequential, none when a rule ends after that element), and the type of action (immediate, extended, sustained). The data are used to obtain a classification of the previously trained neural network, in order to obtain a prediction score for each of the possible values for the "next step" field. These scores are later combined with the results obtained from CPT to return the top 5 best classes. In addition, some filters based on the context are applied, to prevent for example the suggestion of a "trigger" element, if the user has specified "action" as the desired next element. The corresponding blocks are then created and added to the secondary workspace. If the "none" class is present among these top elements, it means that the specific rule element is usually the last element of a sequence, hence a notification of this is added into the "info and help" text area.

4.4 Full rule-based suggestions

Full rule suggestions are obtained starting with a prediction about the next element of the sequence obtained using CPT. The best scoring predictions are added at the end of the rule currently under editing, generating new sequences that end with the candidates for the next element. Each of these sequences is compared to the saved rules using the Jaccard similarity index, to find the rule with the most similar elements for each one. The rules thus obtained are then suggested to the user. We chose to extract a rule for each result from CPT, instead of directly presenting them, to provide more diversified recommendations.

5 User Test

Participants were first contacted via email, and we provided them with a summary of the topic of the evaluation, and a short video (3.30 minutes) illustrating the main functionalities of the visual environment. Eleven people accepted to participate in the test, 3 females and 8 males, with an average age of 30.27 (std. dev. = 4.08), 3 with high school diploma, 6 with a bachelor's degree and 2 with a master's degree. Their previous experiences in use of technologies and programming was evaluated using 1-5 Likert scales. Participants reported a good experience with web usage (average = 4.9), little programming experience (average = 2, std. dev = 1,1), a moderately high interest in new technologies (average = 3.64, std. dev = 0.81). Two of them had already used some tool for defining personalized behaviour based on events (one just to try it out, the other to personalise some behaviour with IFTTT, Google Assistant and the PC).

The user test was carried over via remote calls, where the participants shared their screen, also because the Covid crisis. At the beginning of the call, they could try the environment and ask for clarification about some unclear aspects of the editing process. Then, the tasks list was sent to them and the test started. Each task was carried out individually, and the duration for each one recorded. The tasks were:

1) Compose a rule without using the RS that activates this behaviour: *when the user's heartbeat becomes 150 and user is in the house, activate all the relaxing lights in the house and send an SMS to the caregiver.*
2) Select step-by-step suggestions, then define a rule starting with the "relative position" trigger. The rule must include a compound trigger part (using the "and", "or", "not" operators) and/or a compound action part (using the "sequential" or "parallel" operators).
3) Select full-rule suggestions, then define a rule starting with the "time" trigger. The rule must include a compound trigger part and/or a compound action part.
4) Select a recommendation type of your choice, then define a rule starting with the "rain" trigger. The rule must include a compound trigger part and/or a compound action part. Make sure to switch from the selected recommendation mode to the other during the rule composition.

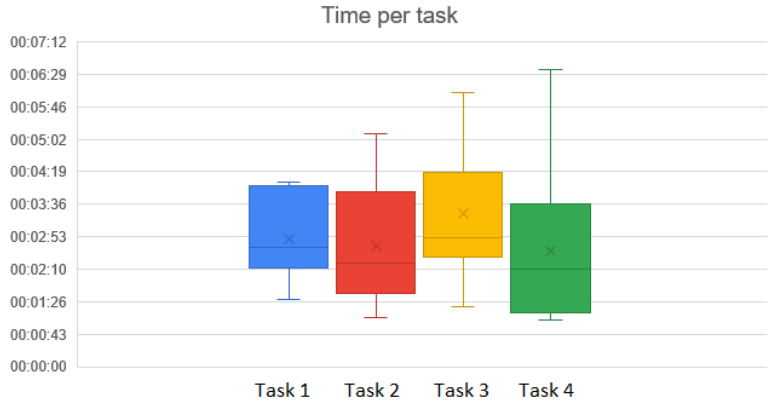

Figure 3: comparison of the timing per task using whisker-and-bar plot.

	Task 1	Task 2	Task 3	Task 4
Average	00:02:49	00:02:41	00:03:23	00:02:33
Max	00:04:05	00:05:10	00:06:04	00:06:36
Min	00:01:30	00:01:05	00:01:20	00:01:02
Std. dev. (seconds)	56,03	81,81	80,62	102,24

Table 2: tabular representation of the timing per task.

The timing of the tasks (see Figure 3) shows that task 3 (complete a rule using full rule suggestions) took more time on average to complete, while the other tasks took lower similar times on average. Moreover, tasks 2, 3 and 4 (which included recommendations) show greater variability. The longer timing on the task 3 can be related to the extra time spent examining the provided rule examples. A post-survey was used to evaluate other usability aspects of the visual editor. All the participants agreed that they preferred to use the editor with a suggestion system enabled. Regarding the recommendations perceived usefulness and relevance with respect to the inserted rule elements, full-rule mode scored on average 3.55 (std. dev. 1.36) while step-by-step on average 4.09 (std. dev. 0.7). With regard to the presentation of the suggestions, full-rules scored on average 3.45 (std. dev. 1.04) and step-by-step 4.27 (std. dev. 0.65). Participants evaluated positively (average 4.18, std. dev 0.6) the usefulness of the possibility to switch between one and the other recommendation mode during rule composition. Also, the block-based approach to defining rules was well received (average 4.36, std. dev. 0.67). Lastly, participants perceived as valid some proposals for further enhancement of the recommendation system, which were considering the user's features (e.g. preferences for a room, device, hour to receive notifications) in the generation of the suggestions (average 4.36, std. dev. 0.67), and the introduction of an explanatory method to allow them to explore why that particular item was suggested (average 3.91, std. dev 1.04).

6 Discussion

From the post-survey and time-per-task data, we note that participants tend to prefer step-by-step suggestions, which also seems to allow for faster composition. However, these data are not conclusive, also considering that the possibility to use both the recommendation methods scored better on the perceived usefulness than the single approaches. Regarded the recorded timings, a longer time is not necessarily a negative aspect, as it can reflect the behaviour of a user who tries to model a rule that really matches his/her interests, instead of being satisfied with a "good enough" rule. We observed this behaviour during the test, where users tried to model rules such as "*send me a reminder when it starts to snow and it is ski season"*, or "*at 2 pm, if I still haven't played today and my emotional state is a bit down, send me a reminder that says 'It's time to play!' and in parallel starts all the activating lights".*

About possible improvements of the RS, various further enhancements can be envisioned. Participants reported that they would appreciate recommendations more tailored to the single user, in order to speed up the rule composition process, to automatically suggest actions based on their activities, and to show possible diverse use cases for a device. Different approaches can be used to define a user profile, e.g. use the created rules or their activations to infer their preferences, or apply a segmentation on the sensor activation data. Data about user profile could be used in the implemented neural network approach, to be able to refine results based also on similarity between users. Another interesting

approach to refine the suggestions is the application of a contextual bandit on the results presented to the user, in a similar fashion to (Intayoad, Kamyod, and Temdee, 2020). This would allow users to act directly on the suggestion list, modelling it to their liking based on the defined contexts. Participants were also favourably disposed to the possibility of including an explanation method, but specified that this should be an optional feature. From their point of view, a graphical tree-based representation of the rule generation process appears intuitively, more suitable and helpful for them. This could be directly applied to the CPT model. Such explanation should also be interactive and illustrate how the results change when the input parameters are modified, following a "what if?" approach (Zürn, Eiband and Buschek, 2020). Still regarding the RS, the focus of this work was on a user-centric perspective, looking at the perception of helpfulness, presentation, and integration in the rule creation workflow. In future work, we aim at further analysing algorithmic metrics such as precision, recall and F-Score.

Concerning the presentation of results, the visualization adopted for step-by-step suggestions appeared to be more convincing than the full-rule one. With full-rule recommendations, the screen area for presenting recommendations was considered too small, and the proposed recommendations too similar to the rule currently under editing. One participant reported that it could be helpful to show some suggestions also for the previous step (e.g. one for the previous and four for the next). Also, some improvements to increase the expressivity of the editor were proposed by participants, for example adding an "else if" operator ("*if it's 7 o'clock and I am in kitchen do this, else if I am in bedroom do that"*). Interestingly, this solution was proposed by a participant without programming experience.

7 Conclusions and Future Work

We propose an approach based on sequence prediction to generate recommendations to give support to end-users during the definition of personalization rules in the IoT context. The paper discusses the design process and aspects that were found useful in designing the two proposed recommendation methods, such discussion can provide useful insights for those interested in introducing intelligent recommendation systems in EUD environments.
The recommendation method was implemented in a block-based tailoring environment, and the results generated and presented using two different modalities, step-by-step and full rule. All the participants in the evaluation user test agreed that the proposed solutions increase the usability of the tailoring environment. Both the presentation methods were well received, with a preference for the step-by-step method. On the other hand, they found that the presentation of full-rule suggestions needs improvements. The possibility of switching between the two modes during the composition of a rule was rated as a highly useful feature. Based on the feedback of the participants to the user test, the design of an approach capable of combining both recommendation modes seem to be a promising direction for further research in this area.
Further improvements on the generation of sequence-based recommendations should consider the user profile and direct user intervention on the suggestion list, which may be supported by some graphical representations also useful for explaining how the recommendations are derived.

Funding

This work was supported by PRIN: PROGETTI DI RICERCA DI RILEVANTE INTERESSE NAZIONALE 2017 – EMPATHY [Grant Number 2017MX9T7H].

REFERENCES

Nesreen K. Ahmed, Amir F. Atiya, Neamat El Gayar, and Hisham El-Shishiny. 2010. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews* 29, no. 5-6 (2010): 594-621.

Barbara Rita Barricelli, and Stefano Valtolina. "A visual language and interactive system for end-user development of internet of things ecosystems." Journal of Visual Languages & Computing 40 (2017): 1-19.

Will Brackenbury, Abhimanyu Deora, Jillian Ritchey, Jason Vallee, Weijia He, Guan Wang, Michael L. Littman, and Blase Ur. 2019. How Users Interpret Bugs in Trigger-Action Programming. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). Association for Computing Machinery, New York, NY, USA, Article Paper 552, 12 pages. https://doi.org/10.1145/3290605.3300782

Federico Cabitza, Daniela Fogli, Rosa Lanzilotti, and Antonio Piccinno. 2017. Rule-based tools for the configuration of ambient intelligence systems: a comparative user study. *Multimedia Tools and Applications* 76, no. 4 (2017): 5221-5241.

Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello, 2019a. Empowering end users in debugging trigger-action rules. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1-13. 2019.

Fulvio Corno, Luigi De Russis, and Alberto Roffarello. 2019b. My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor. 18–33. https://doi.org/10.1007/978-3-030-24781-2_2

Fulvio Corno, Luigi De Russis, and Alberto Roffarello. 2019c. RecRules: Recommending IF-THEN Rules for End-User Development. ACM Transactions on Intelligent Systems and Technology 10 (09 2019), 1–27. https://doi.org/10.1145/3344211

Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2020. HeyTAP: Bridging the Gaps Between Users' Needs and Technology in IF-THEN Rules via Conversation. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pp. 1-9. 2020.

Joëlle Coutaz, and James L. Crowley. "A first-person experience with end-user development for smart homes." IEEE Pervasive Computing 15, no. 2 (2016): 26-39.

Jose Danado and Fabio Paternò. 2012. Puzzle: A Visual-Based Environment for End User Development in Touch-Based Mobile Phones. In Human-Centered Software Engineering, Marco Winckler, Peter Forbrig, and Regina Bernhaupt (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 199–216.

Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. 2017. Specification of Complex Logical Expressions for Task Automation: An EUD Approach. 108–116. https://doi.org/10.1007/978-3-319-58735-6_8

Gerhard Fischer. 2011. Understanding, fostering, and supporting cultures of participation. interactions 18, 3 (May 2011), 42–53. DOI: https://doi.org/10.1145/1962438.1962450

Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Carmen Santoro. 2017. Personalization of Context-Dependent Applications Through Trigger-Action Rules. ACM Transactions on Computer-Human Interaction 24 (04 2017), 1–33. https://doi.org/10.1145/3057861

Ted Gueniche, Philippe Fournier-Viger, Rajeev Raman, and Vincent S. Tseng. "CPT+: Decreasing the time/space complexity of the Compact Prediction Tree." In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 625-636. Springer, Cham, 2015.

Ted Gueniche, Philippe Fournier-Viger, and Vincent S. Tseng. "Compact prediction tree: A lossless model for accurate sequence prediction." In International Conference on Advanced Data Mining and Applications, pp. 177-188. Springer, Berlin, Heidelberg, 2013.

Will Haines, Melinda Gervasio, Aaron Spaulding, and Bart Peintner. "Recommendations for end-user development." In Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI), pp. 42-49. 2010.

Justin Huang and Maya Cakmak. 2015. Supporting Mental Model Accuracy in Trigger-Action Programming. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15). Association for Computing Machinery, New York, NY, USA, 215–225. https://doi.org/10.1145/2750858.2805830

Wacharawan Intayoad, Chayapol Kamyod, and Punnarumol Temdee. 2020. Reinforcement Learning Based on Contextual Bandits for Personalized Online Learning Recommendation Systems. *Wireless Personal Communications* (2020): 1-16.

Hanjo Jeong, Byeonghwa Park, Minwoo Park, Ki-Bong Kim, and Kiseok Choi. 2019. Big data and rule-based recommendation system in Internet of Things. Cluster Computing 22, 1 (01 Jan 2019), 1837–1846. https://doi.org/10.1007/s10586-017-1078-y

Thibaut Le Guilly, Jacob H. Smedegård, Thomas Pedersen, and Arne Skou. 2015. To do and not to do: constrained scenarios for safe smart house. In *2015 International Conference on Intelligent Environments*, pp. 17-24. IEEE, 2015.

Marco Manca, Fabio Paternò, Carmen Santoro, Luca Corcella. Supporting end-user debugging of trigger-action rules for IoT applications, International Journal of Human-Computer Studies, Vol.123, 56-69

Andrea Mattioli, and Fabio Paternò. "A Visual Environment for End-User Creation of IoT Customization Rules with Recommendation Support." In Proceedings of the International Conference on Advanced Visual Interfaces, pp. 1-5. 2020.

Christoph Molnar. *Interpretable Machine Learning*. Lulu. com, 2020.

Alessandro A. Nacci, Vincenzo Rana, Bharathan Balaji, Paola Spoletini, Rajesh Gupta, Donatella Sciuto, and Yuvraj Agarwal. "BuildingRules: A Trigger-Action--Based System to Manage Complex Commercial Buildings." ACM Transactions on Cyber-Physical Systems 2, no. 2 (2018): 1-22.

Bak Nayeon, Byeong-mo Chang, and Kwanghoon Choi. 2018. Smart Block: A Visual Programming Environment for SmartThings. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 32–37. https://doi.org/10.1109/COMPSAC.2018.10199

Elnaz Nouri, Robert Sim, Adam Fourney, and Ryen W. White. "Step-wise recommendation for complex task support." In Proceedings of the 2020 Conference on Human Information Interaction and Retrieval, pp. 203-212. 2020.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. "Sequence-aware recommender systems." ACM Computing Surveys (CSUR) 51, no. 4 (2018): 1-36.

Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. "Personalizing session-based recommendations with hierarchical recurrent neural networks." In proceedings of the Eleventh ACM Conference on Recommender Systems, pp. 130-137. 2017.

Carsten Radeck, Alexander Lorz, Gregor Blichmann, and Klaus Meißner. "Hybrid recommendation of composition knowledge for end user development of mashups." ICIW 12 (2012): 30-33.

Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. *Recommender systems handbook*. Springer, Boston, MA, 2011. 1-35.

Daniel Rough and Aaron Quigley. 2017. Overcoming mental blocks: A blocks-based approach to experience sampling studies. 2017 IEEE Blocks and Beyond Workshop (B&B), 45–48. https://doi.org/10.1109/BLOCKS.2017.8120409

Chowdhury Roy, Soudip, Olexiy Chudnovskyy, Matthias Niederhausen, Stefan Pietschmann, Paul Sharples, Florian Daniel, and Martin Gaedke. "Complementary assistance mechanisms for end user mashup composition." In Proceedings of the 22Nd International Conference on World Wide Web, pp. 269-272. 2013.

Vijay Srinivasan, Christian Koehler, and Hongxia Jin. 2018. RuleSelector: Selecting conditional action rules from user behavior patterns. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, no. 1 (2018): 1-34.

Andrew Stratton, Chris Bates, and Andy Dearden. 2017. Quando: Enabling Museum and Art Gallery Practitioners to Develop Interactive Digital Exhibits. In End-User Development, Simone Barbosa, Panos Markopoulos, Fabio Paternò, Simone Stumpf, and Stefano Valtolina (Eds.). Springer International Publishing, Cham, 100–107.

Maryam Tavakol, and Ulf Brefeld. "Factored MDPs for detecting topics of user sessions." In Proceedings of the 8th ACM Conference on Recommender Systems, pp. 33-40. 2014.

Matúš Tomlein, Sudershan Boovaraghavan, Yuvraj Agarwal, and Anind K. Dey. 2017. CharIoT: an end-user programming environment for the IoT. In *Proceedings of the Seventh International Conference on the Internet of Things*, pp. 1-2. 2017.

Bartłomiej Twardowski, "Modelling contextual information in session-aware recommender systems with neural networks." In Proceedings of the 10th ACM Conference on Recommender Systems, pp. 273-276. 2016.

Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). Association for Computing Machinery, New York, NY, USA, 3227–3231. https://doi.org/10.1145/2858036.2858556

Beidou Wang, Xin Guo, Martin Ester, Ziyu Guan, Bandeep Singh, Yu Zhu, Jiajun Bu, and Deng Cai. 2018. Device-Aware Rule Recommendation for the Internet of Things. InProceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18). Association for Computing Machinery, New York, NY, USA, 2037–2045. https://doi.org/10.1145/3269206.3272009

David Weintrop. 2019. Block-Based Programming in Computer Science Education. Commun. ACM 62, 8 (July 2019), 22–25. https://doi.org/10.1145/3341221

David Weintrop, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David Shepherd, and Diana Franklin. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. 1–12. https://doi.org/10.1145/3173574.3173940

Lefan Zhang, Weijia He, Jesse Martinez, Noah Brackenbury, Shan Lu, and Blase Ur. "AutoTap: synthesizing and repairing trigger-action programs using LTL properties." In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), pp. 281-291. IEEE, 2019.

Martin Zürn, Malin Eiband, and Daniel Buschek. 2020. What if? Interaction with Recommendations. In *ExSS-ATEC@ IUI*. 2020.