Personalised Recommendations for Daily Automations Controlled by Mobile Augmented Reality

ANDREA MATTIOLI, CNR-ISTI, Italy
FABIO PATERNÒ, CNR-ISTI, Italy

Internet of Things-enabled home automation is starting to significantly transform our daily lives. Users must be able to configure and coordinate the connected objects in their dwellings to personalise and fully benefit from their potentialities. Trigger-action programming is one relevant approach to enable users to create useful automations. However, current approaches mainly based on visual Web or mobile interfaces have limitations, such as the difficulty in finding and selecting the correct object and associated services. Mobile augmented reality is an interaction modality that can support more direct and usable automation control. In this perspective, the support of a recommendation system can facilitate users in creating personalised automations. This paper presents a system for generating personalised daily automations recommendations and presenting them in a mobile augmented reality solution in order to facilitate their monitoring and creation. Seven classification approaches were assessed on different datasets to determine their ability to provide personalised and context-aware recommendations. We also report a user study (N = 16) showing that the personalised recommendation system improves user performance when creating automations in a trigger action format with a mobile augmented reality editing environment.

CCS Concepts: • **Human-centered computing** → **Mixed / augmented reality**; **Ubiquitous and mobile computing**; • **Information systems** → **Recommender systems**.

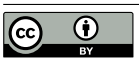
Additional Key Words and Phrases: Mobile Augmented Reality, Internet of Things, Trigger-Action Programming, Home Automation, Recommender Systems

ACM Reference Format:
Andrea Mattioli and Fabio Paternò. 2025. Personalised Recommendations for Daily Automations Controlled by Mobile Augmented Reality. *Proc. ACM Hum.-Comput. Interact.* 9, 4, Article EICS016 (June 2025), 22 pages. https://doi.org/10.1145/3734863

1 Introduction

Nowadays, Internet-connected objects are increasingly present in our daily lives, making the Internet of Things (IoT) paradigm a widespread reality. Another increasingly present aspect is artificial intelligence, which is employed in various areas, including home automation. Using these technologies without considering their usability diminishes people's control over their objects and devices and ultimately limits their benefits. One relevant approach to allow human control in an environment rich in IoT objects is trigger-action programming (TAP). It is an End-user Development (EUD) programming method to define automation rules, e.g. "When this change is detected in the environment, activate this object", which has proven to be effective for enabling objects, devices, and services to operate in a coordinated manner [5, 47]. TAP has been used in several research tools [14, 17, 20] and commercial software (e.g. IFTTT, Zapier, SmartThings). These tools are mainly

Authors' Contact Information: Andrea Mattioli, CNR-ISTI, HIIS Laboratory, Pisa, Italy, andrea.mattioli@isti.cnr.it; Fabio Paternò, CNR-ISTI, HIIS Laboratory, Pisa, Italy, fabio.paterno@isti.cnr.it.



ACM 2573-0142/2025/6-ARTEICS016
https://doi.org/10.1145/3734863

based on the visual paradigm, using metaphors (such as puzzle blocks or pipes) or information organisations (e.g. trees) to ease the creation of automations.

However, some aspects of these visual programming environments are still problematic, for example, selecting the correct services since these platforms usually adopt a vendor-centric abstraction [12], the limited expressiveness of the rules that these tools can create [47, 48], or the configuration of the timing aspects of a trigger [5, 24] (when an automation should activate). For such reasons, there is growing interest in researching new approaches more direct and interactive to creating automations, such as conversational agents or augmented reality (AR). The use of AR in smart homes is particularly relevant [4, 25], as it makes the connection between various objects in the environment and their representation within the application more immediate and direct. Using MAR (Mobile Augmented Reality), the user can create automations through the personal smartphone while moving in her environment by framing an object, configuring one service, and then moving to other objects. MAR allows for a fast and intuitive selection of the required object compared to the visual editors that use lists or functionality hierarchies. This is obtained by exploiting a commonly used device, such as the smartphone, without requiring specific visors or headsets. Thus, it can be largely adopted when targeting daily automations, such as in the smart home context. For example, there may be dozens of different smart lights in a home, and the user may not remember the name of the specific light or find it challenging to select the correct object and functionality in a visual tool [23]. MAR makes this connection evident. In this context, the introduction of Recommendation Systems (RSs) can further help users as they can, for instance, suggest rules similar to those made by a user or provide pre-configured parts of automation to complement the behaviour that the user is defining. The RS can be integrated to further facilitate this configuration and the continuation of the automation creation, generating suggestions when the user approaches an object (or interacts with an AR visualisation) and when she moves away. Bridging the gap between RSs and MAR platforms for IoT environments can be a fruitful approach to enhance the possibilities of dynamically monitoring and creating automations these platforms offer. However, as we discuss in the related work section, little work has been dedicated to this topic, with limited approaches. In this paper, we present and assess a novel mobile AR-based smart home platform to create TAP rules integrated with a recommendation system, intending to make it easier for end users to personalise the automated behaviours of their environments. The proposal's novelty consists of using specific strategies for both the rule creation part (allowing the creation of automations composed of multiple conditions and actions in MAR) and the recommendation part (using a context-aware approach to generate step-by-step suggestions, also considering the textual part of user input using a language model). The most innovative contribution lies in an approach that organically combines MAR and RS by considering both the functional implementation of the system and user interaction. The research questions that drove the study are:

- **RQ1:** Does integrating a recommendation system in a mobile augmented reality tool simplify the creation of automations?
- **RQ2:** Does the integration of a recommendation system within a MAR tool prevent errors during the automation creation?

2 Related Work

While some work considering mobile augmented reality for controlling smart homes has been put forward [2, 23, 41], their integration with a recommender system has not yet been considered so far. Relevant literature regarding recommendations for smart home automations or similar applications has considered different approaches. An overview is presented by Felfernig and colleagues [16], concluding that directly applying collaborative filtering can be used for simple cases such as

suggesting an automation compatible with the user's IoT gateway. For more complex use cases, a sequential pattern mining approach is proposed. However, no metric assessment or user study was reported.

2.1 Recommendations from logs of user-generated events

One approach to IoT recommendations is to generate them from the actual use of the devices or environments detected by the sensors. RuleSelector [43] is a tool that allows the interactive selection of automations generated from user behaviour detected through a smartphone. It is intended to minimise the problem of rule mining algorithms generating too many automations by introducing specific metrics. One approach to synthesising TAP rules from logs of sensor events detections and manual object activations is Trace2TAP [55]. The algorithm uses a SAT-solver and symbolic reasoning to generate the rules, which can also handle unordered events. Then, a ranking/clustering system is used to sort the resulting rules. Another approach that leverages user-generated event sequences is proposed by Song and colleagues [42]. In this work, the sequences of user actions are exploited to predict the next actions the user may take at the service level and present them as a recommendation. A limitation of these approaches is that automations do not only concern predictable routine events, which can be learned directly by examining activation logs, but also sporadic events. Given their non-regular nature, they are extremely difficult to learn and, consequently, to be used in these recommendations.

2.2 Recommendations from automations dataset

Another relevant strategy is to generate recommendations starting from existing rules datasets. RecRules [10] is an approach to recommend automations based on their purposes. An algorithm is used to enrich automations with collaborative and semantic information. Next, a semantic graph is constructed, and path-based features are used to train a learning-to-rank model and obtain the top-n recommendations. An approach based on similar principles is rtar[52]. It uses a hybrid model that contains a rule collaboration graph (which defines users, rules, devices/services and their collaborative relations) and a functionality hierarchy (which defines triggers and actions hierarchies organised at three abstraction levels). Kuang and colleagues [26] proposed an approach to create rule recommendations considering collaborative information among users. The RS models the similarity between users using graphs representing the objects in their installations. The graphs are then converted into a vector representation of users and rules using graph contrastive learning. The described approaches exploit datasets of automations consisting of simple rules containing one trigger and one action. As current commercial and research platforms for TAP automations allow for the creation of more advanced automations, these approaches are not fully adapted to the current smart home automation landscape. Another approach [30] examines recommendations mainly from their presentation's perspective. Since automations comprise multiple rule elements, they can be presented in a fragmented (step-by-step suggestion) form or as a whole rule. Both approaches can be valid for recommendation in this context, but users preferred rule parts recommendations, perceiving them as more guiding.

2.3 Use of natural language descriptions in recommendations

Some works have investigated using natural language descriptions of IoT automations to create new rules. Dalal and Galbraith [13] evaluated three seq2seq architectures, observing that these models could learn If-Then rules exploiting the various linguistic aspects found in their natural language descriptions. More recently, Yusuf and colleagues [54] proposed RecipeGen, a seq2seq approach that turns an input description into a sequence of triggers and actions. The output's granularity level is the object/external entity (such as weather API) reference and service for that object/entity.

Unlike our study, this paper presents a recommendation model without the end-to-end system implementation and user evaluation. Another approach that uses natural language is HeyTAP [11]. It is a conversational TAP rule-making platform that outputs rule recommendations through user interaction and information related to connected objects. Users can specify personalisation intents and high-level preferences, such as sustainability or security, used as semantic filters to recommend relevant rules. Although user interaction with the platform is described, the implementation of the underlying recommendation system is not detailed and metric evaluation is missing. A recent related contribution is TAP-TAG [51]. It is a machine learning model that captures and integrates a TAP rule's structural and textual parts. The structural branch uses a Knowledge Graph embedding model, projecting information extracted from the rules into embedding. The textual branch employs a CNN module that uses the vectors generated by Word2Vec. Then, a multi-modal representation fusion component jointly trains the parameters of the two models, fusing a rule's structural and semantic (textual) embeddings. The final embeddings can be used for recommendation tasks. Unlike our contribution, that work focuses on the data representation of TAP rules and not on the actual integration of a TAP recommender system in a platform for home automation management, and it does not consider the user in the recommendation generation. Furthermore, the presented approaches only support single-trigger single-action rules. An initial study to address the issues in RS for IoT automations through AR [31] proposes recommendations for completing the automation currently being created exploiting Doc2Vec. The RS uses the partial rule already entered by the user to suggest a completion also considering the configuration of triggers or actions entered, such as the associated values or text. However, the generated recommendations are not personalised but the same for all users, and thus, it acts more like an "autocomplete" than a proper recommendation system. Furthermore, in the reported user test, the recommendations received mixed feedback, as they were perceived as useful but significantly increased the composition time and did not improve satisfaction in using the app.

2.4 AR/MAR in the IoT

To develop the rule creation tool, we analysed how AR and MAR has been used in IoT settings. The typical use is to display additional information [2], either placed on top of a specific object or via generic overlays not related to a specific object [29, 34, 37]. Reality Editor [23] is one of the earliest tools that introduced the use of MAR for controlling and linking the behaviour of IoT devices. It acts as a virtual remote by overlaying interactive controls like buttons or knobs directly onto the physical devices or relocating the interface to another space. Users can also visually connect functions between devices using "virtual tags," enabling interactions such as activating one device when another is turned on. Additionally, it supports remote control by relocating the visual interface of a device to another space. Instead of extending the interface, our goal is to support users in specifying multi-object automations. HoloFlows [41] introduces an AR platform for designing IoT workflows using a head-mounted display, drawing inspiration from Business Process Modelling. It allows users to link devices using virtual "wires" and create automated behaviours using an event-condition-action format. The projected holograms help users interact with objects in their surroundings. Compared to traditional modelling tools like Camunda and NodeRED, HoloFlows offers a more intuitive experience with less technical complexity. However, it requires a headset and focuses on connecting nearby objects. MagiPlay [44] is an educational AR game designed to teach children computational thinking. It uses immersive 3D environments to let players capture and use virtual objects to build basic IoT automations using if-then logic, with support for logical operators like OR and AND. The focus is strictly on education, not on real-world automation tasks. ARticulate [9] helps users discover the functions of unfamiliar smart objects in a space through AR visual cues and a chat-based interface. Inspired by apps like Snapchat, it

highlights interactive objects and enables control via a simple voice/text assistant enhanced by an autocomplete. However, it does not support creating or managing automations between multiple devices. Spontaneous Automation Control [2] is a mobile app that uses a smartphone's camera to recognise smart devices or rooms, overlaying contextual information on the screen. It supports the creation of basic automations (single trigger and one action) by guiding the user through the setup. Triggers are based on recognised objects, and actions are manually selected afterwards. Our approach allows more expressive rules, e.g., including multiple conditions. From this perspective, we provide an original solution that integrates RS with MAR, supporting users in dynamically creating trigger-action automations that can also have multiple triggers and/or actions.

2.5 Gaps in existing research and commercial applications

From this analysis of related work, it emerges that integrating recommendations with AR systems for IoT is still a novel possibility which should be further explored. Furthermore, RSs considering compound automations beyond the simple one-trigger one-action structure are still under-utilized and need better solutions. It is also important to consider that existing commercial trigger-action platforms, even the ones allowing the definition of complex automations such as Home Assistant, do not include recommendation support to facilitate input or limit errors. Another notable aspect is that almost all studies on RS in IoT only address the algorithmic aspect of the problem without considering how to integrate RS in a way that is useful for those who interact with these systems. A system that provides 'good' recommendations must help users create automations more conveniently and limit errors. For this reason, we implemented an end-to-end platform comprising a specific recommendation system for MAR and a rule creation app to test it. The evaluation of this system considered both the underlying machine learning models and the interaction of users with the application.

3 Design of the proposed solution

3.1 Key terms

The key terms used in the paper are detailed here for clarity. A "Rule" (or automation) is a sequence of "Rule elements" consisting of at least one "Trigger" and one "Action." Usually, in the trigger part, there is an "Event," which corresponds to a change of state that initiates the checking of the rule activation, and there can be one or more "Conditions", which act as filters to specify better when the actions should be triggered. The action part can consist of one or more object or service activations, for example, "turn on the light" and "send me a notification". The term "Rule element type" refers to the distinction of each rule element in event, condition, or action type. By "Service", we refer to the specific functionality of an object (for instance, reading the temperature from a sensor) or an application (such as getting the weather forecasts for the next 24 hours) that is needed to define a "Rule element". For example, in the Rule element *"element-ecatype : event; trigger : kitchenTemperatureSensor-getTemperature; operator : EQUAL; value : 20"* the service is *kitchenTemperatureSensor-getTemperature*. Please note that this representation can be easily translated into the natural language sentence "When (event) the kitchen temperature becomes 20 degrees" through the composition of its elements.

3.2 Requirements

Starting from the literature review, we identified a list of requirements for a TAP recommender system in MAR:

1 The RS should model the user/functionality relation, considering the concrete possibilities of the available objects. Modelling the user/item relation is fundamental to providing

relevant recommendations. A finer granularity can be desirable since each object or external application can exhibit diverse functionalities. Therefore, functionality will be used as the primary element for identifying relevance for the user. In addition, it is also useful to consider the values and operators contained within each rule element. The corresponding textual description of a rule can be encoded using transformers [13, 54].

2 It should be possible to allow the optional insertion of high-level features. High-level features [11, 19] can be considered to make the recommendations more relevant and to provide the user with more control. Additional information, such as the goal of a rule, helps determine which services will be part of it and how they can be configured. These features, associated with rules through tags, can act as intermediate semantic elements between items [49] (e.g., "both of these automations are tagged with energy saving"), thus facilitating understandable explanations [45] through a "what-if" [28] approach (How do the recommendations change if I change this tag?). The interactive insertion of high-level tags allows people to see recommendations beyond those they usually receive without tinkering with the RS settings. For instance, user insertion of goals has been successfully implemented in GoalPods [27], leading to recommendations outside the user's usual "information bubble" and, at the same time, accurate.

3 The RS should consider the user behaviour in different phases (approaching an object and while editing a corresponding rule) and provide editable rule elements to support the completion of the rule. One feature of AR is the ability to exploit the physical distance of the person from the objects of interest. This spatial interaction is an aspect "naturally" enabled by AR and exploited by the RS, which would be cumbersome to apply with a traditional mobile app. Using AR, the user can create automations while moving in her environment by framing an object, configuring one service, and then moving to other objects. Thus, the RS should generate recommendations when the user approaches an object, e.g., within a distance below a threshold or when selecting the object of interest (before the configuration, or pre-recs). Another moment to present recommendations is when the interaction with an object ends, and the user continues configuring other rule elements (post-recs). For both situations, it is relevant to present recommendations as rule elements instead of complete rules ("step-by-step"), as they can be more easily integrated with the rule in editing [30]. The step-by-step suggestions should streamline the rule creation process as a sort of multistep wizard [39]. However, the user must be free to modify the received recommendation, using them as a starting point template to model the desired behaviour [40].

4 The RS should consider the rule element type (Events, Conditions, Actions). Previous work indicates how the ECA structure suits the IoT context [3, 18] and makes it possible to express the personalised behaviours people expect. However, it also reveals how the temporal aspects related to this structure, particularly the distinction between events and conditions, can confuse users [5]. The RS must, therefore, support users in this choice by highlighting the type of rule element best suited for the situation.

3.3 MAR Rule Creation App

The implemented MAR app to create automation rules is a Unity app which uses the ARFoundation library to manage the interactions with the scene. The app is conceptually based on the *ContextData* and *AnchorCreator* modules. *ContextData* contains the code to retrieve descriptions of objects that can be interacted with in the environment. At startup, these descriptions are retrieved, and Game Objects are created with the information needed to interact with them (e.g., type of device and coordinates). The *AnchorCreator* module then accesses these Game Objects. It creates visualisations that will be inserted into the environment, creates AR anchors and assigns them to each visualisation to fix their position. The positioning of visualisations is enabled by identifying, at the first use, where they should be placed using a specific functionality of the app. This functionality captures

the coordinates of a position in the environment in relation to a fixed point (e.g., an image target that the user can place wherever she prefers: we used one image per room). During later use, it is sufficient to detect that image to place all the visualisations in a room.

We did not use object detection, as it can cause issues with our use case (e.g., how to distinguish a "smart" lamp from a standard one, between objects with almost the same shape, or detect small sensors from afar).

3.4 The Proposed Recommendation Solution

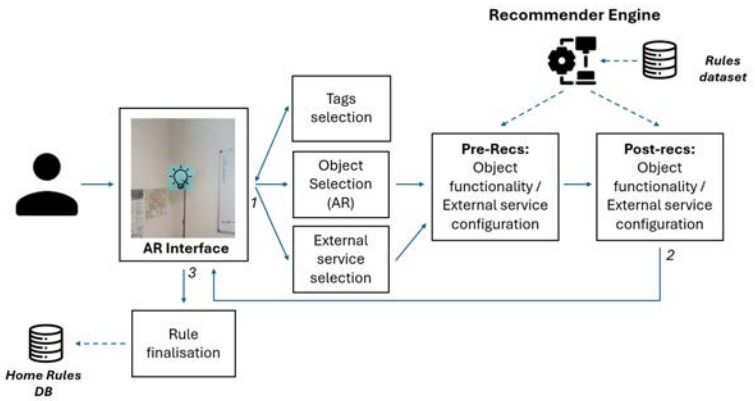


Fig. 1. Overall functioning of the proposed solution

The proposed solution is an RS that, at the beginning of the rule creation, asks the user to (optionally) select some tags (Req. 2) concerning the intended goal (e.g., comfort, wellbeing, security), the user type (caregiver, family), and the more broad situation to which the rule is designed for (summer, school season, going out). Then, according to the recommendation phase (pre-configuration recommendations or post-configuration), a quick metric is used to select the candidate rule elements (Req. 3), considering the services more relevant for the phase and the inserted tags. Afterwards, it computes their scores employing a more powerful model (Req. 1). Finally, a set of heuristics is used to present the results considering the type of rule elements (events, conditions, and actions) already present in the rule (Req. 4). Figure 1 shows the scheme of the main interaction (solid arrows) and data (dotted arrows) flows in the proposed solution. First, from the AR interface, the user can select a tag, an object, or an external service (1). Then, she configures the rule element using pre- and post-recs (2). The view goes back to the AR interface, where the user can again configure objects or external services or save the finalised version when the rule is complete (3).

In general, creating automations with the app is done by moving around in the real environment and selecting via the augmented reality view the object to be used in the rule. This opens a mobile-style panel in which the desired "rule element" can be configured. The user can add this rule element to the rule currently being in editing, and save the automation if it is complete (at least one trigger and one action are present) or continue this process by selecting other objects and configuring other rule elements. Functionalities not associated with physical objects are instead called via a dedicated button in the interface. In order to assess the effectiveness of the RS introduction, the application was developed in two variants, with and without recommendations. The interaction with both is described below.

3.5 App without recommendations

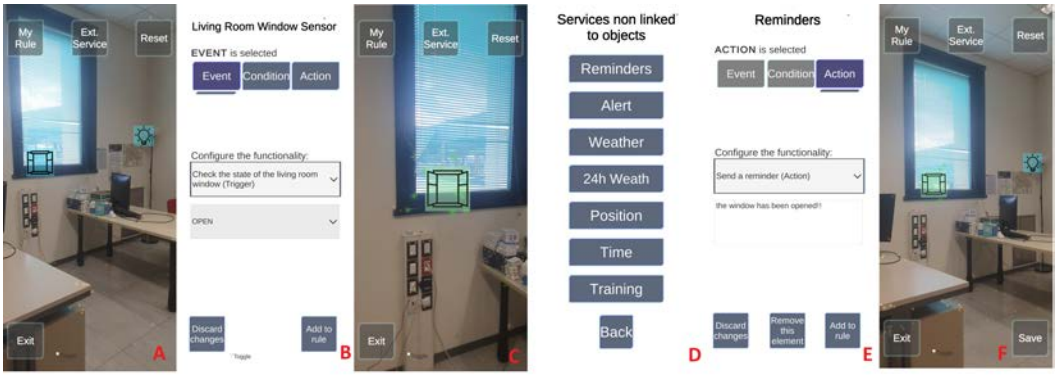


Fig. 2. The process of creating an automation with the app without recommendations.

An example of interaction with the basic application is shown in Figure 2. When the user approaches the living room window (Figure 2-a) she wants to create an automation that alerts when it is opened. She then activates the possibility of editing an automation involving this connected object (Figure 2-b). For this purpose, she selects the ”Event” rule part and defines the event by selecting the desired functionality and value in the “rule configuration” screen. After pressing ”Add to rule,” the app returns to the camera view, showing the augmented reality view (Figure 2-c). The user continues by selecting the ”External Services” button, which activates the visualisation of the list of services available, from which the user can select the desired ”Reminders” service (Figure 2-d). Then, she can enter the message in a “service configuration” screen (Figure 2-e). After adding this Rule Element, the app returns to the camera view, also showing the ”Save” button (Figure 2-f) since both a trigger and an action have been inserted.

3.6 App with recommendations

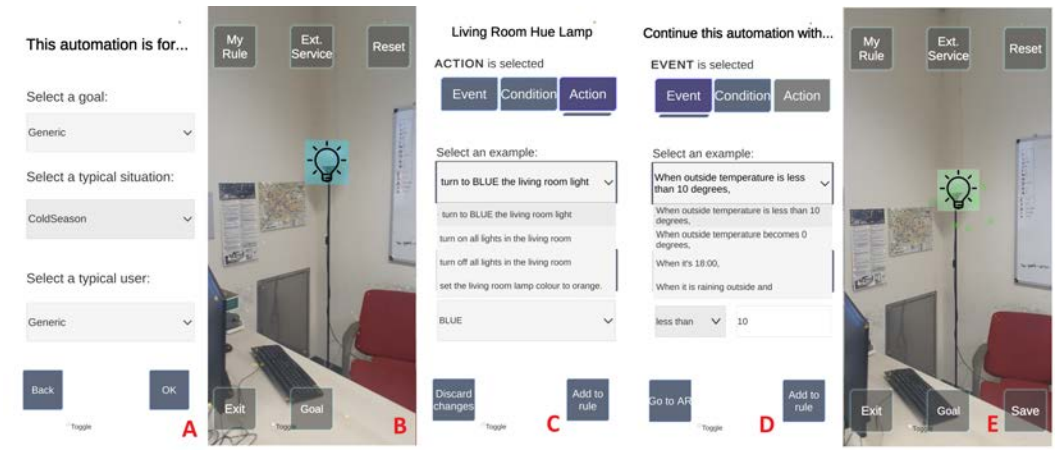


Fig. 3. The process of creating an automation with the app with recommendations.

Figure 3 shows a concrete example of user interaction with the recommendation functionalities of the app. At the beginning, the user can insert a "tag" for the goal, typical user and situation that the rule she is about to define targets (Figure 3-a). After selecting the tags (or skipping this screen), the user starts the rule editing, approaching a lamp (Figure 3-b). After selecting the associated visualisation, a list of recommendations is shown ("pre-rec"). The user can accept a recommendation as it is, accept it and then modify it according to her needs, or ignore it and continue creating the rule as she wishes. In this case, she selects the possibility of setting the light colour to blue (Figure 3-c). Then, the RS generates a list of "post-recs" recommendations to continue the definition of the automation. Since the partially inserted rule comprises only one action, the most suitable rule type element for continuing the automation is "Event". Hence, the list of event recommendations is presented by default. Among these, the first results are related to low temperatures since the user selected the "ColdSeason" situation (Figure 3-d). After clicking "add to rule", the app switches back to the camera view (Figure 3-e), where the user can continue the automation by selecting other objects and configuring other "Rule elements" (for instance, adding a condition), or if the automation is potentially (as in this case) complete press the button to save it. In the example, the presence in the first positions of recommendations related to the Situation tag "Cold Season" is due to the fact that the user left the default "Generic" tag for Goal and User Type. Hence, the suggestions with the Situation tag "Cold Season" (the first two) are prioritised.

4 Recommendation engine

4.1 Recommendation pipeline

The implemented RS is based on a Flask API, and it receives as inputs the user identifier, a list of the services for the selected object or application, the rule elements already configured in the automation (e.g., *eca-type : action; action : notifications-smsNotification; value : "remember to take the umbrella!"*), a sentence describing the partial rule entered by the user ("send me the notification remember to take the umbrella‼"), three lists containing the tags eventually selected by the user at the beginning of the rule creation, and a flag indicating the type of recommendation required (pre or post). It outputs a list of rule elements, representing configurations for the object the user is approaching (pre) or a list of possible continuations for the rule (post). A scheme of its functioning is in Figure 4. After receiving the data from the AR rule editor it selects, at the "pre" stage, the rules containing the services of the object the user approached, and in the "post" phase, the ones that include the services present in the partial rule entered by the user. Next, the resulting rules are sorted using the TF-IDF metric, by comparing user-entered tags with those of the rules. This step allows users to enter "short-term interests" other than those for which they usually create automations. For example, a user who normally enters energy-saving oriented automations may sporadically want to target comfort instead. Using this multi-stage approach [35], recommendation candidates are obtained, thus passing less input to the deep learning model and speeding up the RS. After applying the TF-IDF metric, another filter removes from the candidate rules those elements not needed at this stage of the recommendations (i.e., rule elements not for the object of interest in "pre-recs" and rule elements involving services already used in the partial rule inserted by the user). Please note that this is needed because we are now reasoning at the rule element level, whilst in the previous step, we were at the rule level. The remaining elements are passed to a recommendation model, and the new relevance score is used to sort them. The last step is to generate three ECA lists (one for events, one for conditions, and one for actions) and propose these lists with a priority that considers the type of the rule elements already inserted (for instance, if an action is inserted, priority is first events list, then the conditions list, and finally the actions list).

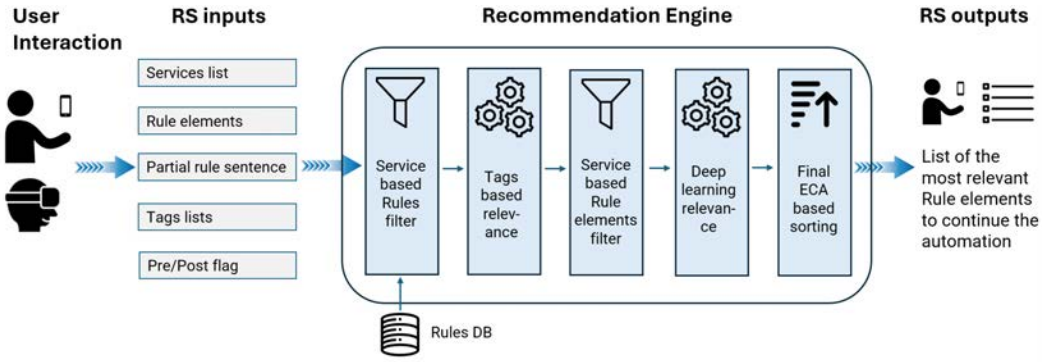


Fig. 4. Overview of the recommendation pipeline functioning.

4.2 Classification models

To identify a suitable classification model for the recommendation engine, we tested seven models, five taken from the literature and two modifications of these models. Their output is a score between 0 and 1, representing how well the input features (e.g., a user ID and an object service) fit together. We used Neural Collaborative Filtering and BERT models because they are widely used and obtained good results in different classification tasks, also when involving TAP rules [7]. We also assessed the Logistic Regression and the Random Forest models, as they are widely adopted and suited for binary classification considering multiple factors (user, item, and context).

Logistic Regression: Logistic Regression is a basic statistical model used to predict the probability of an event with a binary outcome. It applies the logistic or sigmoid function to a linear combination of input features, mapping the predicted values to a probability distribution between 0 and 1. Logistic regression is a simple and robust model used in countless applications, which makes it a good baseline approach.

Random Forest classifier: Random Forest[6] is an ensemble method which averages the output of multiple decision tree predictors, each trained on a random subset of the data. The random forest algorithm uses bagging and feature randomness techniques to create an uncorrelated forest of decision trees, hence reducing overfitting and increasing the overall precision of the predictions.

Collaborative Filtering (CF): Collaborative Filtering is a widespread approach to recommendations. It assumes that users with similar tastes will prefer similar items, leveraging the preferences of a community of users to generate recommendations. In model-based approaches, users and items are represented by an embedding vector. In the implemented model the prediction, which is the score of an item for a user, is obtained through the dot product between the vector representations of the user and the item.

Neural Collaborative Filtering (NCF): NCF [22] models the relationship between users and objects using a neural architecture that can learn an arbitrary function from the data instead of the inner product as in classical approaches. NCF uses two paths to model the user-item relationship: in the former, the element-wise product of input vectors is performed. In the second, this relationship is modelled by a standard neural network using a common tower architecture. To provide maximum flexibility for the models, they learn embeddings separately and are fused by concatenating their last hidden layers.

BERT Classifier: BERT is a pre-trained language model that considers the relationships of tokens in the sentence and computes word embeddings [15]. After the embedding layer, two layers were added, a fully connected one and another with a single neuron to output the score. Then,

we fine-tune the model with input sentences structured in the following format: *"The user " + userIdentifier + " used the context " + previouslyInsertedRuleElements + " with the rule element " + ruleElement.*

Contextual Neural Collaborative Filtering (C-NCF): NCF input layers consist of feature vectors describing users and objects, which can be customised to support the modelling of different relationships [22]. Thus, we encoded the services of the rule elements already entered by the user in the rule being created using a binary vector with "1" assigned to these features. This input was used in the two NCF model paths described above, in the first by running the dot product between this embedding and the user embedding and then between the result and the item embedding. In the second, concatenating it before entering the dense neural network (similarly to [46]).

Contextual Neural Collaborative Filtering with BERT (C-NCF BERT): This approach (see Figure 5) merges the structure of C-NCF with that of BERT by concatenating in the last hidden layer of C-NCF the fully connected layer of BERT (hence excluding the last classification neuron).

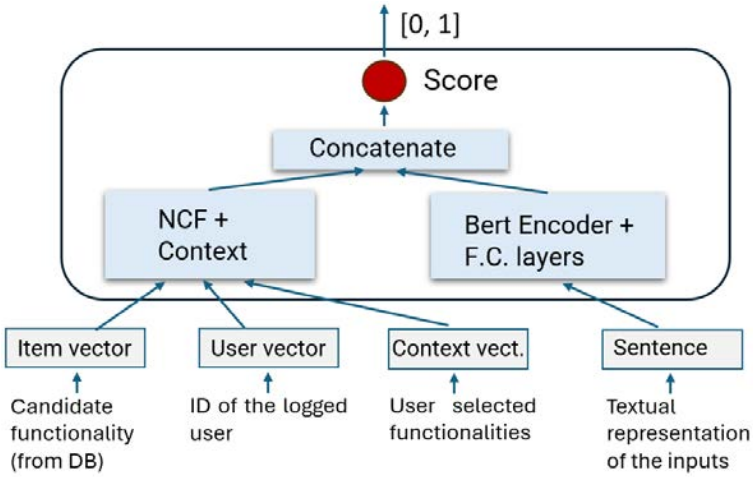


Fig. 5. Outline of the model C-NCF with Bert.

4.3 Dataset

We built a dataset to evaluate these models, starting from a TAP rules dataset publicly available on GitHub [21]. The dataset includes diverse automation structures, enabling recommendations for flexible automation platforms. Other existing datasets feature only simple one-trigger one-action automations [33, 48] or omit pseudo-ids of the rule authors [53], making them less useful for personalised recommendations. The dataset consists of 434 automations already deconstructed into 1292 Rule Elements. The automations were created by 77 different users and involved 169 services. The dataset was expanded using a sliding window approach. For instance, if the rule is of length three, there will be three windows: one with the first part as "contextual information", having the second one as the target; one with the second as context, having the third as the target; finally, one with the first and second as context, and the third as the target. The process was also performed with sliding windows of size one, thus containing only the "target" but without the "context". In this way, the dataset contains scenarios where the user approaches an object without having configured anything yet, and where the user has already entered one or more rule elements. Rules were also analysed in reverse order, thus doubling the examples. The resulting dataset has

Table 1. F1/AUC scores for the models training

	Bert	C-NCF Bert	C-NCF	NCF	CF	Random Forest	Logistic Reg.
1 Neg 0 Pos	0.833/0.916	0.848/0.924	0.902/0.95	0.907/0.941	0.903/0.943	0.863/0.93	0.73/0.82
1 Neg 1 Pos	0.934/0.946	0.933/0.945	0.95/0.965	0.946/0.945	0.948/0.953	0.938/0.967	0.84/0.829
5 Neg 5 Pos	0.988/0.995	0.986/0.994	0.977/0.993	0.937/0.952	0.938/0.957	0.988/0.997	0.765/0.839
Average	0.918/0.952	0.922/0.954	0.943/0.969	0.93/0.946	0.93/0.951	0.929/0.964	0.778/0.829
IFTTT	0.775/0.858	0.812/0.885	0.867/0.982	0.821/0.974	0.695/0.911	0.836/0.972	0.608/0.903

5450 positive examples but no negative ones, making classification arduous. Therefore, random negative sampling was applied to the dataset, creating new ”negative” rules by replacing an element of the original one (e.g. the action with its configuration) with another element randomly retrieved from the dataset (ensuring that it was not the same functionality present in the original rule). To get a more general idea of the performance of the models, we created variations of the dataset with different sample numbers (only a negative sample, one negative and a positive duplicate, and five negatives with five positive duplicates)[1]. Finally, the rules were labelled with tags to be used with the recommendation pipeline. By tag, we mean a representative instance referring to a high-level feature. Based on previous work (e.g., [11, 19, 32]), we identified the User Type (e.g. including the tags Caregiver and Student), Goal (e.g., Comfort, Energy Saving) and Situation (e.g., Nobody at home, Cold Season) features. Then, we mapped the automations with the corresponding tags.

4.4 Models training and accuracy assessment

Model training was done in a Python environment using Keras and Scikit-Learn. The values of the hyperparameters were kept the same in the NCF, C-NCF, C-NCF Bert, and Bert models (binary cross-entropy as loss function, 40 epochs, patience set to two on the validation loss, batch size 8, optimiser Adam) except for the learning rate, which was set to 0.01 for the NCF and NCF-C models, and 2e-5 for Bert and NCF-C Bert ones, as we observed better results using these values. The network structures common among the models have also been kept the same. For the Random Forest model, we kept the default Scikit-Learn parameter (n_estimators= 100, Gini impurity to measure the quality of a split, max_depth = none, min_samples_split = 2). The results of the 5-fold cross-validation on the three datasets (F1 and AUC scores) are in Table 1. It can be observed that the C-NCF model is the most robust, performing well under all conditions.

4.5 Scalability assessment

We performed a scalability test to assess whether these results would change in a real-world situation with many users and rules. For this test, we used the last version of the IFTTT dataset from Mi and colleagues [33]. The dataset contains 279828 rules, each with one trigger and one action, created by 121207 users. The triggers and actions are characterised by the channel name (e.g., Dropbox) and the specific functionality (e.g., save file to Dropbox). We considered the concatenation of these two descriptions as the recommendation’s target item (Dropbox-save file to Dropbox). There are a total of 1131 different target triggers and 743 target actions. The sliding window strategy was used in this case as well. From each rule, four data entries were created: trigger as items to suggest, action as context; trigger as items to suggest with context = ”none”; action to suggest with the trigger as context; action to suggest with ”none” as context. This process results in 1119312

[1]The resulting expanded datasets and the code for the models training and assessment (including the IFTTT scalability test) are available at https://github.com/andrematt/MARRS/

input data. The lack of negative feedback was handled directly in the training loop where, for each sample, negative ones were generated using random negative sampling, and a different weight (alpha level) was assigned to them[38]. The results of this assessment are in the last line of Table 1. The C-NCF model performed well even in this setup.

5 User Study

To validate the effectiveness of the recommendation pipeline and, in general, the inclusion of RS in the tailoring environment, we performed a user test comparing the version of the app without recommendations (baseline) and with recommendations. As a classification model, we used the C-NCF, as it performed best in the metric assessments.

5.1 Participants

We recruited sixteen participants (7 females, 9 males) through direct email or mailing lists. There was no compensation for participation. Their ages ranged from 22 to 40 years (average = 30.31, std. dev. = 5.08). In general, they reported some proficiency with programming (3 reported no programming experience, 2 had low experience, 3 had medium experience, 4 had good, and 4 had very good experience). Seven of them reported experience in using automations to control smart environments. Among the platforms used, they indicated Alexa Routines (4), IFTTT (3), Home Assistant (2), Google Home (2), Smart Life (1), and Phillips HUE (1). 3 reported having used augmented reality on smartphones, using Pokemon Go (2) and applications made in Unity (1).

5.2 Test organisation

The study took place in a laboratory with two rooms simulating, respectively, a living room and a bedroom. In each room, there were four smart objects: an air purifier, multifunctional sensor (humidity, temperature, motion, amount of light), window sensor and actuator, and smart light for the living room; sleep tracking sensor, multifunctional sensor and actuator for the window, and smart light for the bedroom. In addition, it was possible to select seven services not directly linked to objects (see Figure 3-d).

The test was organised in 8 phases: general introduction, familiarisation with one version of the app, task performance, questionnaire for the used version, familiarisation with the other version of the app, task performance, questionnaire, and final questionnaire for demographic information. Before starting, participants signed an informed consent. After the study, participants stayed further to comment and discuss their experiences. The introduction phase served to give a brief idea of TAP programming and to acquaint participants with the environments in which the tasks would take place. Then, they could try the version of the application they were assigned as initial (with or without recommendations). The order in which the applications (installed on a Samsung Galaxy S10) were used was reversed for each user to counterbalance the learning effect. Questionnaires for both versions included the 10 statements of the SUS test and a statement regarding the perceived ease of creating automations with this version of the application. The questionnaire for the app with recommendations also included four statements specific to this system adapted from the framework for user-centric evaluation of RS proposed by Pu and Chen [36], namely:

- **S1** I found it helpful to be able to use the recommendations while performing the tasks.
- **S2** The system gave me good recommendations.
- **S3** The recommendations shown made me more confident in my choice.
- **S4** Overall, I am satisfied with the recommendation system.

In this context, a ”good recommendation” means a recommendation that is relevant to completing the automation that the user is creating. The statements correspond to the dimensions of perceived

usefulness, recommendation quality, decision confidence, and satisfaction. They had to be assessed using a 5-point Likert scale where 1 corresponded to "strongly disagree" and 5 to "strongly agree". At the end of the questionnaires, two open questions asked them to elaborate on what aspect of the application they appreciated the most or the least and why.

Tasks: The study was set up as a within-subject. Participants had to complete eight tasks (the complete list is in Appendix A), corresponding to the creation of as many automations. In the beginning, if they were using the app with recommendations, they had to select the goal indicated for that group of tasks. The tasks were divided into two groups, the first four to be done with one version of the application and the second four with the other. In both groups, the first two automations to be created were simple (i.e. an event and an action), and the last two were composite (in one case, an event, two conditions, and an action; in the other, an event, a condition, and two actions), hence, the difficulty of the tasks was comparable. For each task, the time taken to complete it, the errors made, and any other notes or mental processes expressed by the participants were recorded.

5.3 Results

Time to complete the tasks: The times to complete tasks (see Figure 6) indicated that, in general, users were faster using the app with recommendations (M = 52.36 s, std. dev. = 30.48 s) than with the basic version (M = 75.89 s, std. dev. = 27.57 s). This difference is significant (Shapiro-Wilk normality test W = 1, p = 0.084, Student's t-test t = 7.2, p = <0.001, effect size = 0.9). We examined the times to complete compound and simple tasks separately to form a better idea. Regarding simple tasks, using recommendations (M = 32.8 s, std. dev. = 18.7 s) has led to a faster completion rate compared to the base version (M = 62.47 s, std. dev. = 25.56 s). The data were found not normally distributed, as suggested by the presence of outliers. The difference between the two distributions was significant (Shapiro-Wilk normality test W = 0.9, p = 0.03, Wilcoxon signed-rank W = 7, p = <0.001, effect size = 1). Considering only data from compound tasks, again using recommendations (M = 71.9 s, std. dev. = 27.39 s) resulted in faster tasks than using the base version (M = 89.3 s, std. dev. = 22.78 s). Data was found to be normally distributed, and the difference was significant (Shapiro-Wilk normality test W = 1, p = 0.8, Student's t-test t = 3.9, p = <0.001, effect size = 0.7).

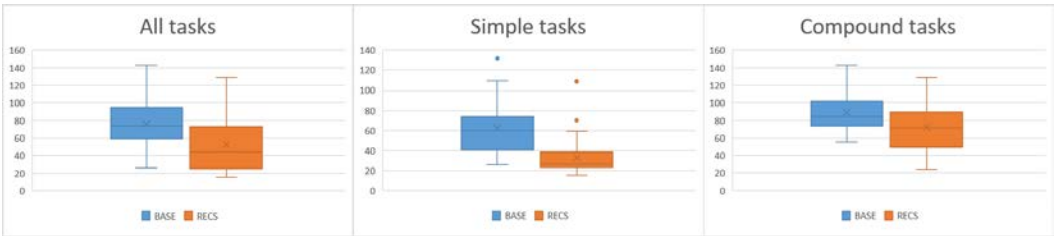


Fig. 6. Time per task.

Errors: We identified the errors by comparing the automations required by the task with the ones produced by participants. The error counting reveals how they concentrate in the basic application (N = 22, mean = 1.37, median = 1) while they are lower in the version with recommendations (N = 4, mean = 0.25, median = 0). In the basic version, the prevalent error was not correctly choosing the events and conditions to model the trigger (N = 15). Other errors found were selecting the wrong operator for the rule element (3 cases), wrong service (2 cases), wrong value (1 case), and not having entered a rule element (1 case). In the version with recommendations, the errors were, in one case, inserting an event instead of a condition and, in one case, adding two more actions

to a rule (counted as two errors), and in one case, inserting an action different from the required one. These data are not normally distributed (Shapiro-Wilk, W = 0.9, P = 0.036), and the Wilcoxon Signed Rank indicates significance (W = 3.1, p = <0.002, effect size = 1).

SUS responses: Overall, the SUS scores indicate that the platforms have good usability, with a preference for the version with recommendations (mean score: 82,3, min=55, max=100, std. dev=12.2) over the basic version (mean score: 78,1, min=42.5, max=92.5, std.dev= 13.89). The Shapiro–Wilk test confirms that the data are normally distributed (Shapiro-Wilk normality test, W=0.9, p=0.4), but the difference between the distribution is not significant (Student's t-test t = -1.8, p = 0.087).

Other statements: Responses to the statements about the perceived ease of using the application were generally positive for both the basic version (mean = 4, median = 4) and the version with recommendations (mean = 4, median = 4.37). Questions specific to the recommendation system gathered strong positive feedback (Q1: mean = 4.5, median = 5; Q2: mean = 4.6, median = 5; Q3: mean = 4.5, median = 5; Q4: mean = 4.56, median = 5).

Comments associated with open questions: The open-ended responses were labelled and grouped, identifying themes that emerged inductively from them. Below are the resulting themes, together with some representative quotes.

Positive aspects: For the *basic* version of the application, several features stood out as particularly appreciated. One of the most frequently mentioned strengths (4 occurrences) was the rule recap feature, which clearly summarizes the events, conditions, and actions configured within an automation. Users found this helpful, as reflected in comments such as: "The my rule screen with my settings summarised" and "The summary of the rules gives more clarity on what you have set. Very useful". Another key positive aspect was the intuitiveness of the rule creation process (4), which was described as simple and effective: "The ease with which you can set the rules and the usefulness in home automation" and "The ease of creating an automation". The application's overall interface was also praised for its clarity and ease of use (3). Users appreciated its intuitive graphics, with one remarking, "As I used the app, I felt more confident in using it," while another noted, "The fact that there are no examples seems to streamline the interface and thus a simplicity of use emerges". Additional features that received positive feedback included the AR visualizations and visual feedback (2), which allowed users to "see the location of sensors" and provided a "visual feedback of a sensor currently involved in an automation." The interactive AR experience was another highlight (2), with users valuing the ability to "move in the environment to set what I need" and appreciating that "the application works by pointing at the objects of interest" Lastly, the possibility of creating compound automations was noted as an advantage (1): "The possibility of combining several conditions to initiate an action".

In the version of the application that includes *recommendations*, the most appreciated aspect was the recommendation system itself (9), which was praised for simplifying interactions and improving efficiency. As one participant put it, "The recommendation system makes the interaction simpler and more immediate for the user compared to the other version." Another noted, "I really appreciated the suggestions the app gave me step by step, I was able to create rules much faster and more easily." A third highlighted its seamless integration: "The suggestion system is well integrated into the system and also very useful makes the creation of even complex rules much faster". Other positive aspects included the option to set optional tags, which allowed users to adapt automations to different needs (2) "The possibility of setting the goal to adapt to different needs", and the AR visualizations along with the graphic effect on activation (2): "The visualisation and the icons used for the devices are very nice," and "the fact that the objects such as the dehumidifier after creating an action or condition turn green". Additional appreciated features were the "Continue this automation with…" screen (1), which enabled users to extend a rule without selecting a new

device, the automatic pre-selection of the rule element type (event/condition/action) (1), and the flexibility to create automation in any preferred order rather than following a strict path (1).

Negative aspects: Despite these positive aspects, users identified areas that could be improved. In the *basic* version, one critique was the graphic style of the menu (3), which some found immature: "Graphical appearance of the menu" and "The graphical style is still a bit immature". Another issue was the insufficient differentiation in input types (2), with users suggesting improvements such as a dedicated clock for time selection: "The time selection could be improved by inserting a clock etc." and "I would differentiate the input types based on the choices (e.g. for a time an input of type time, for a date of type date etc)". Some users also struggled with understanding the conceptual distinction between events and conditions (2): "Difficult to understand the differences between event and condition, with a few more trials you can definitely understand better". There were also calls for more noticeable visual feedback (2): "More feedback or visualisation of the sensors in the other rooms in a different colour" and "Greater differentiation between the three phases when selecting the action of the object concerned". Other areas requiring attention included the lack of dedicated space in the AR environment for services without a corresponding object (1): "I would expect either a kind of virtual notice board or positioned sensors (e.g. weather outside the window)", the positioning of some UI elements such as the exit button (1), and missing objects, specifically a door sensor (1).

For the version with *recommendations*, similar concerns were raised regarding the UI style and button placement (4). Users felt the interface could be more visually appealing and consistent: "The style of the user interface can be improved to make it more attractive," "Always the graphics of the menus," and "The style of the buttons in service is not consistent with the interface in home". One user also highlighted an issue where they unintentionally clicked 'exit' and had to start over. There were also suggestions for more explicit visual feedback (2), particularly when selecting recommendations or interacting with UI elements: "I would have appreciated the inclusion of some visual element that would indicate to me the activation/deactivation of it" and "I think there is a need for more explicit feedback when selecting elements, saving etc.". Additionally, users noted that recommendation tags could be better integrated and highlighted (2): "The case studies ('in case of rain', for example) tend to be separate from other commands" and "When it gives you a recommendation it should add at the end (Wellbeing)". Lastly, the integration of recommendations into the main menu was mentioned as an area for enhancement (1): "The integration of the recommendations in the interface where you define events etc."

6 Discussion

Overall, the response from participants was positive, albeit highlighting some areas for improvement and further investigation. The various feedback from users and the classification metrics assessment of the machine learning models will be analyzed below in more detail and with reference to the defined research questions. Some broader implications for design that can be generalised from our study are also reported.

RQ1: Does integrating a recommendation system in a mobile augmented reality tool simplify the creation of automations?

From the results of the SUS test, we can see that although the platform received good perceived usability scores both with and without recommendations, the use of recommendations improved this aspect. The time taken by users to complete the tasks indicates that, in general, the use of recommendations sped up the creation of automations. The difference appears significant in simple and compound automations, but is more marked in simple automations. From statements S1-S4, the one concerning the ease of creating automations with the two versions, and from the responses to open questions, it emerges that the recommendations were one of the most

appreciated aspects of the application, made the participants more confident during the use, and were considered relevant for the configuration tasks. From observing the participants during the task completion, no particular difficulty in using the recommendation system emerged. On the contrary, the approach based on configuring one "rule element" at a time, supported by recommendations before and after the insertion, was usually grasped during the familiarization phase before starting the tasks **(Implication 1: present single rule elements recommendations is a viable strategy).** Additional considerations can be made about the components of the recommendation pipeline. The selection of the user goal did not confuse participants and was cited as a preferred feature by two of them. For instance, one participant, who has a professional interest in caregiving, stated that switching between general user and caregiver through tags would be extremely useful. Still, she would need a longitudinal study to assess this feature entirely. Another consideration concerns the "ECA sorting". Some users closely followed the event-condition-action order, selecting the services in sequence to complete the rule. Others instead preferred to start from the objects in the AR view, regardless of their position in the tasks. For the users of the first group, the order of the recommendations proposed by the "ECA sorting" was sometimes suboptimal. For instance, one task required them to create a rule with multiple conditions, but after the insertion of a condition, the system gave priority to actions to conclude the automation. This did not cause serious inconvenience because users could manually select the type of rule element to display suggestions for or discard them and continue the automation manually. In any case, a system that learns the user's event/condition/action selection preferences and orders recommendations accordingly would be a useful addition to the RS. This can be supported with a classifier which, based on the rules already entered by the user, learns the preferred insertions style and presents the recommendations accordingly **(Implication 2: the selection order preferred by users when creating automations should be considered)**. Overall, the proposed approach resulted suitable for supporting rule creation in MAR. One possible improvement that has emerged from participants' comments is to give more evident feedback when a recommendation is modified, making it clearer that the modified suggestion will be included in the rule, not the original one. It is also to be noted that one participant explicitly reported, during the use, her preference for the version without recommendations. She stated that it appeared leaner and, hence, easier to use.

RQ2: Does the integration of a recommendation system within a MAR tool prevent errors during the automation creation?

From the results of the study, we can see that the inclusion of the recommendation system significantly decreased the errors made during the tasks, and particularly those regarding the choice between event and condition. This is particularly relevant, as the distinction between events and conditions is a crucial source of ambiguity in TAP [5, 24], and is essential to clearly express this concept to help users shape a correct mental model of the functioning of the automated system. We witnessed this challenge during the test when participants often thought aloud about the duration of triggers to choose between events and conditions **(Implication 3: recommendations can also help users to disambiguate between events and conditions)**. For example, one participant had doubts about how to model "when I sleep for more than 8 hours" because although the event of exceeding 8 hours is instantaneous, the triggering action of sleeping extends for a prolonged period. We can associate the decrease of the event/condition selection errors while using the RS with two factors: the event/condition/action suggestion based on heuristics, which "pre-selected" the most suitable type of rule element, and the textual examples, which reported the natural language description of the rule element. For instance, the event rule element type was often introduced by the "When" keywords and the condition by the "If", depending on the terms used in the textual description of the automations used for the recommendation. These, in combination with the "My Rule" screen that explicitly divides the rule into its events/conditions and actions, led to the

alleviation of this problem. This seems to suggest that using different keywords to denote events and states leads to a more correct mental model of TAP rules by users **(Implication 4: visual and linguistic cues can be used in combination with recommendations to assist users further with the event/condition distinction)**, although which terms are most appropriate is currently being studied[1]. It should be noted, however, that the "My Rule" panel was also present in the version without recommendations Furthermore, one participant reported the need, at least initially, for an even more explicit representation of the distinction between events and conditions. He suggested an animation that illustrates that first an event must be triggered, and then the system proceeds to verify the conditions.

Another aspect worth discussing concerns the **metric assessment of the models**. The results tell us that the intuition of modelling together the user-item relation, the previously inserted rule elements as the "context", and the textual description of the automation is suitable but not optimal in all conditions. The models that also use the textual description (Bert and Context-NCF Bert) performed well in the dataset version extended with 5 positive and 5 negative examples, but in the other cases, they did worse than simpler models such as C-NCF or random forest. The Bert model performed relatively poorly in the scalability test on the IFTTT dataset, an indication that the model is not completely capable of capturing the user/item relationship without using specific encoders. In contrast, models that can encode the context separately have performed well in most cases, particularly Context-NCF. This indicates that the use of context together with a user and item representation leads to a correct interpretation of the next element to be included in the rule without the need for the textual component to clarify it further **(Implication 5: text-based approaches to recommendations are feasible in this setting, but the user-functionality relation and the context emerge as more important factors).** Another possible cause for the worse performance of Bert in the IFTTT dataset is the less detailed description of the automations present compared to the multiple rule elements dataset. To wrap up, C-NCF and to a lesser extent random forest performed well both in a small dataset containing automations with multiple rule elements, as well as in a large-scale one with 1-trigger 1-action automations. This indicates the possible effectiveness of the models in different real-use situations. This result is also relevant to RS implementations for IoT that do not use augmented reality since the functioning of the core models is not tightly coupled with that of the rest of the recommendation pipeline.

7 Conclusions and Future Work

In this paper, we present the design and implementation of a RS for automations in a mobile augmented reality application for smart homes. It considers the specificities of the automations, such as the ECA structure, the partitioning of recommendations into Rule elements, and the different stages in which to provide them. The models used as the core of the RS were evaluated by metrics, obtaining good results. The model that performed best in this respect (C-NCF) was included in the recommendation pipeline. A user test was carried out with a TAP platform with this system and without recommendations. The results indicated better usability of the platform with recommendations, which also reduced errors in the creation of automations. Regarding the limitations of this work, it must be reported that the user study considered only one of the models assessed from the metric standpoint. Furthermore, a more extended evaluation should be done in a real home installation to better verify the usefulness of the recommendation system over a long period of time, its ability to adapt to the user, and the effectiveness of the tags to explore automations other than those normally shown. Another possibly limiting aspect concerns the latency that the recommendation pipeline could introduce into the system. During the user test, we did not notice any particular lag introduced by the RS. However, it may be present in systems using larger datasets like the IFTTT one. To prevent this issue, we specifically defined the RS as a

multistep system, where a quick metric (TF-IDF) is used to select the candidates to be sorted by the more powerful (and slower) deep learning model. Even if this is useful for decreasing possible lag issues, further investigation on this aspect is needed.

In future work, we will conduct a longitudinal user study to better understand the strengths and weaknesses of the designed RS. Other aspects we intend to explore are the explainability of recommendations, for example by showing why a recommendation has been proposed. We are also planning to explore how to personalise the system further, for instance, by automatically inferring the user goal [50] and by considering additional information such as user models and personality traits [8].

Acknowledgments

This work has been partly supported by the Italian MUR PRIN 2022 PNRR Project P2022YR9B7, End-User Development of Automations for Explainable Green Smart Homes, funded by European Union - Next Generation EU. This work is also partly funded by European Union - Next Generation EU, in the context of The National Recovery and Resilience Plan, Investment 1.5 Ecosystems of Innovation, Project Tuscany Health Ecosystem (THE), CUP: B83C22003920001.

References

[1] Margherita Andrao, Barbara Treccani, and Massimo Zancanaro. 2023. Language and Temporal Aspects: A Qualitative Study on Trigger Interpretation in Trigger-Action Rules. In *International Symposium on End User Development*. Springer, 84–103.

[2] Raffaele Ariano, Marco Manca, Fabio Paternò, and Carmen Santoro. 2023. Smartphone-based augmented reality for end-user creation of home automations. *Behaviour & Information Technology* 42, 1 (2023), 124–140.

[3] Nayeon Bak, Byeong-Mo Chang, and Kwanghoon Choi. 2020. Smart Block: A visual block language and its programming environment for IoT. *Journal of Computer Languages* 60 (2020), 100999.

[4] Andrea Bellucci, Paloma Diaz, Alvaro Montero, Ignacio Aedo, and Pablo Lopez Coya. 2022. "See with the app's eyes what your real eyes cannot see": design potentials for Mobile Augmented Reality at home. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*. 1–5.

[5] 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*. 1–12.

[6] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[7] Bernardo Breve, Gaetano Cimino, Vincenzo Deufemia, Annunziata Elefante, et al. 2023. A BERT-based Model for Semantic Consistency Checking of Automation Rules (S).. In *DMSVIVA*. 87–93.

[8] Federica Cena, Cristina Gena, Claudio Mattutino, Michele Mioli, Barbara Treccani, Fabiana Vernero, and Massimo Zancanaro. 2022. Incorporating personality traits in user modeling for EUD. *arXiv preprint arXiv:2207.03999* (2022).

[9] Meghan Clark, Mark W Newman, and Prabal Dutta. 2022. ARticulate: one-shot interactions with intelligent assistants in unfamiliar smart spaces using augmented reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 1 (2022), 1–24.

[10] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. RecRules: recommending IF-THEN rules for end-user development. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 5 (2019), 1–27.

[11] 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*. 1–9.

[12] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2022. How do end-users program the Internet of Things? *Behaviour & Information Technology* 41, 9 (2022), 1865–1887.

[13] Dhairya Dalal and Byron V Galbraith. 2020. Evaluating sequence-to-sequence learning models for if-then program synthesis. *arXiv preprint arXiv:2002.03485* (2020).

[14] Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. 2017. Empowering end users to customize their smart environments: model, composition paradigms, and domain-specific tools. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 2 (2017), 1–52.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[16] Alexander Felfernig, Seda Polat-Erdeniz, Christoph Uran, Stefan Reiterer, Muesluem Atas, Thi Ngoc Trang Tran, Paolo Azzoni, Csaba Kiraly, and Koustabh Dolui. 2019. An overview of recommender systems in the internet of things. *Journal of Intelligent Information Systems* 52, 2 (2019), 285–309.

[17] Daniela Fogli, Matteo Peroni, and Claudia Stefini. 2017. ImAtHome: Making trigger-action programming easy and fun. *Journal of Visual Languages & Computing* 42 (2017), 60–75.

[18] Daniela Fogli, Matteo Peroni, Claudia Stefini, et al. 2016. Smart home control through unwitting trigger-action programming. In *Proc. 22nd Conf. Distrib. Multimedia Syst.(DMS)*. 194–201.

[19] Mathias Funk, Lin Lin Chen, Shao Wen Yang, and Yen Kuang Chen. 2018. Addressing the need to capture scenarios, intentions and preferences: Interactive intentional programming in the smart home. *International Journal of Design* 12, 1 (2018), 53–66.

[20] 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 (TOCHI)* 24, 2 (2017), 1–33.

[21] github.com. 2024. TAREME dataset. Retrieved May 17, 2024 from https://github.com/andrematt/trigger_action_rules/

[22] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[23] Valentin Heun, James Hobin, and Pattie Maes. 2013. Reality editor: programming smarter objects. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. 307–310.

[24] 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*. 215–225.

[25] Pascal Knierim, Paweł W Woźniak, Yomna Abdelrahman, and Albrecht Schmidt. 2019. Exploring the potential of augmented reality in domestic environments. In *Proceedings of the 21st International Conference on human-computer interaction with mobile devices and services*. 1–12.

[26] Zhejun Kuang, Xingbo Xiong, Gang Wu, Feng Wang, Jian Zhao, and Dawen Sun. 2024. A Recommendation System for Trigger–Action Programming Rules via Graph Contrastive Learning. *Sensors* 24, 18 (2024), 6151.

[27] Yu Liang, Aditya Ponnada, Paul Lamere, and Nediyana Daskalova. 2023. Enabling goal-focused exploration of podcasts in interactive recommender systems. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*. 142–155.

[28] Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–15.

[29] J Antonio Garcia Macias, Jorge Alvarez-Lozano, Paul Estrada, and Edgardo Aviles Lopez. 2011. Browsing the internet of things with sentient visors. *Computer* 44, 5 (2011), 46–52.

[30] Andrea Mattioli and Fabio Paternò. 2021. Recommendations for creating trigger-action rules in a block-based environment. *Behaviour & Information Technology* 40, 10 (2021), 1024–1034.

[31] Andrea Mattioli and Fabio Paternò. 2023. A mobile augmented reality app for creating, controlling, recommending automations in smart homes. *Proceedings of the ACM on Human-Computer Interaction* 7, MHCI (2023), 1–22.

[32] Andrea Mattioli and Fabio Paternò. 2023. Understanding User Needs in Smart Homes and How to Fulfil Them. In *International Symposium on End User Development*. Springer, 125–142.

[33] Xianghang Mi, Feng Qian, Ying Zhang, and XiaoFeng Wang. 2017. An empirical characterization of IFTTT: ecosystem, usage, and performance. In *Proceedings of the 2017 Internet Measurement Conference*. 398–404.

[34] Georgios Mylonas, Christos Triantafyllis, and Dimitrios Amaxilatis. 2019. An augmented reality prototype for supporting IoT-based educational activities for energy-efficient school buildings. *Electronic Notes in Theoretical Computer Science* 343 (2019), 89–101.

[35] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[36] Pearl Pu, Li Chen, and Rong Hu. 2011. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*. 157–164.

[37] Jannish A Purmaissur, Praveer Towakel, Shivanand P Guness, Amar Seeam, and Xavier A Bellekens. 2018. Augmented-reality computer-vision assisted disaggregated energy monitoring and iot control platform. In *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*. IEEE, 1–6.

[38] Steffen Rendle. 2021. Item recommendation from implicit feedback. In *Recommender Systems Handbook*. Springer, 143–171.

[39] Audrey Sanctorum and Brenda Ordoñez Lujan. 2024. Towards End-User-Driven Generation of IoT Applications. In *Companion Proceedings of the 16th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 66–73.

[40] Audrey Sanctorum, Luka Rukonic, and Beat Signer. 2021. Design requirements for recommendations in end-user user interface design. In *International Symposium on End User Development*. Springer, 204–212.

[41] Ronny Seiger, Romina Kühn, Mandy Korzetz, and Uwe Aßmann. 2021. HoloFlows: modelling of processes for the Internet of Things in mixed reality. *Software and Systems Modeling* 20, 5 (2021), 1465–1489.
[42] Yunpeng Song, Yiheng Bian, Xiaorui Wang, and Zhongmin Cai. 2024. Learning from User-driven Events to Generate Automation Sequences. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 4 (2024), 1–22.
[43] Vijay Srinivasan, Christian Koehler, and Hongxia Jin. 2018. RuleSelector: Selecting conditional action rules from user behavior patterns. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–34.
[44] Evropi Stefanidi, Dimitrios Arampatzis, Asterios Leonidis, Maria Korozi, Margherita Antona, and George Papagiannakis. 2020. Magiplay: An augmented reality serious game allowing children to program intelligent environments. *Transactions on Computational Science XXXVII: Special Issue on Computer Graphics* (2020), 144–169.
[45] Nava Tintarev and Judith Masthoff. 2012. Beyond explaining single item recommendations. In *Recommender Systems Handbook*. Springer, 711–756.
[46] Moshe Unger, Alexander Tuzhilin, and Amit Livne. 2020. Context-aware recommendations based on deep learning frameworks. *ACM Transactions on Management Information Systems (TMIS)* 11, 2 (2020), 1–15.
[47] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. 2014. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 803–812.
[48] 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*. 3227–3231.
[49] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th international conference on Intelligent user interfaces*. 47–56.
[50] Gang Wu, Liang Hu, Yuxiao Hu, Yongheng Xing, and Feng Wang. 2025. User intention prediction for trigger-action programming rule using multi-view representation learning. *Expert Systems with Applications* 267 (2025), 126198.
[51] Gang Wu, Ming Wang, and Feng Wang. 2024. TAP with ease: A generic recommendation system for trigger-action programming based on multi-model representation learning. *Applied Soft Computing* (2024), 112163.
[52] Qinyue Wu, Beijun Shen, and Yuting Chen. 2020. Learning to recommend trigger-action rules for end-user development: A knowledge graph based approach. In *International Conference on Software and Software Reuse*. Springer, 190–207.
[53] Haoxiang Yu, Jie Hua, and Christine Julien. 2021. Analysis of ifttt recipes to study how humans use internet-of-things (iot) devices. In *Proceedings of the 19th ACM conference on embedded networked sensor systems*. 537–541.
[54] Imam Nur Bani Yusuf, Lingxiao Jiang, and David Lo. 2022. Accurate generation of trigger-action programs with domain-adapted sequence-to-sequence learning. In *Proceedings of the 30th IEEE/ACM international conference on program comprehension*. 99–110.
[55] Lefan Zhang, Weijia He, Olivia Morkved, Valerie Zhao, Michael L Littman, Shan Lu, and Blase Ur. 2020. Trace2tap: Synthesizing trigger-action programs from traces of behavior. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–26.

A Appendix

Scenario 1: Environmental management

You want to configure some automations to better organize your home environment and simplify its management (and also avoid waste due to forgetfulness), automate lights and windows and send messages to notify you of non-optimal situations.

If the application version allows it, enter the goal **organizing**. Then, define the following automations:

- At 6 pm, set the light in the living room to a warm colour (orange).
- When I leave the house, turn off all the lights in the living room.
- If the air quality in the living room becomes poor, I am in the room and the living room window is closed, turn on the air purifier.
- When it starts raining and the living room window is open, change the light in the living room to red and send a reminder message.

Scenario 2: Wellbeing

You want to create automations to improve your well-being (particularly regarding sleep) and help you motivate yourself to study/work and do physical activity.

If the application version allows it, enter the **wellbeing** goal. Then, define the following automations:

- When the humidity in the bedroom becomes above 80%, it sends an alert notification.
- When I sleep for more than 8 hours (regardless of the time), it sends a notification to remind me to study/work.
- When the temperature in the bedroom becomes below 18 degrees and if the window is open and I am in bed, close the window.
- At 6.25 pm, if I've taken fewer than 5,000 steps that day, it sends a notification suggesting I go for a walk and changes the colour of the bedroom light to yellow.