From Users' Intentions to IF-THEN Rules in the Internet of Things

From Users' Intentions to IF-THEN Rules in the Internet of Things

FULVIO CORNO, Politecnico di Torino
LUIGI DE RUSSIS, Politecnico di Torino
ALBERTO MONGE ROFFARELLO, Politecnico di Torino

In the Internet of Things era, users are willing to personalize the joint behavior of their connected entities, i.e., smart devices and online service, by means of trigger-action rules such as "IF the entrance *Nest* security camera detects a movement, THEN blink the *Philips Hue* lamp in the kitchen." Unfortunately, the spread of new supported technologies make the number of possible combinations between triggers and actions continuously growing, thus motivating the need of assisting users in discovering new rules and functionality, e.g., through recommendation techniques. To this end, we present $HeyTAP^2$, a semantic Conversational Search and Recommendation (CSR) system able to suggest pertinent IF-THEN rules that can be easily deployed in different contexts starting from an abstract user's need. By exploiting a conversational agent, the user can communicate her current personalization intention by specifying a set of functionality at a high level, e.g., to decrease the temperature of a room when she left it. Stemming from this input, $HeyTAP^2$ implements a semantic recommendation process that takes into account *a*) the current user's *intention*, *b*) the connected entities owned by the user, and *c*) the user's *long-term preferences* revealed by her profile. If not satisfied with the suggestions, the user can converse with the system to provide further feedback, i.e., a *short-term preference*, thus allowing $HeyTAP^2$ to provide refined recommendations that better align with the her original intention. We evaluate $HeyTAP^2$ by running different offline experiments with simulated users and real-world data. First, we test the recommendation process in different configurations, and we show that recommendation accuracy and similarity with target items increase as the interaction between the algorithm and the user proceeds. Then, we compare $HeyTAP^2$ with other similar baseline recommender systems. Results are promising and demonstrate the effectiveness of $HeyTAP^2$ in recommending IF-THEN rules that satisfy the current personalization intention of the user.

CCS Concepts: • **Information systems** → **Recommender systems**; **Retrieval tasks and goals**; **Users and interactive retrieval**; • **Human-centered computing** → **Natural language interfaces**; Interactive systems and tools; • **Theory of computation** → *Semantics and reasoning*; *Abstraction.*

Additional Key Words and Phrases: Trigger-Action Programming, Abstraction, Conversational Recommender System, Semantic Web, Internet of Things, Functionality

ACM Reference Format:
Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2020. From Users' Intentions to IF-THEN Rules in the Internet of Things. *ACM Transactions on Information Systems* 0, 0, Article 000 (2020), 33 pages. https://doi.org/10.1145/1122445.1122456

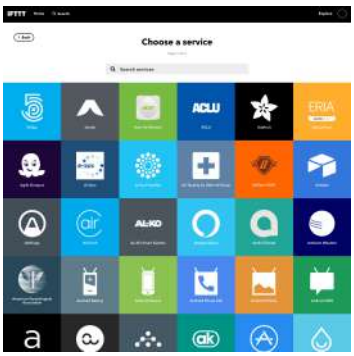
Authors' addresses: Fulvio Corno, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, Italy, 10129, fulvio.corno@polito.it; Luigi De Russis, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, Italy, 10129, luigi.derussis@polito.it; Alberto Monge Roffarello, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, Italy, 10129, alberto.monge@polito.it.

1046-8188/2020/0-ART000 $15.00
https://doi.org/10.1145/1122445.1122456

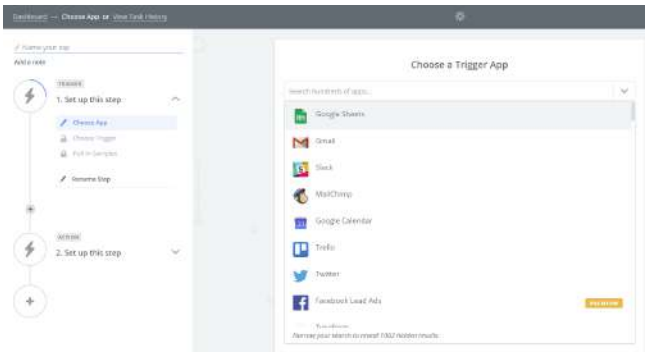
1 INTRODUCTION

The number of people using the Internet has passed 4.5 billion marks in January, 2020, with more than 3.8 billion people actively using online services such as messaging platforms and social media [9]. In the Internet of Things (IoT) era, everyday objects are no longer disconnected from the Internet and they can be controlled remotely [39]. The advent of the IoT already helps society in many different ways [17]: people can interact with a multitude of "smart" devices, ranging from lamps and thermostats in their homes, to building management systems in their workplaces. As a result, users can easily access a complex network of *connected entities*, be they smart devices or online services, that can communicate with each other, with humans, and with the environment.

In this scenario, End-User Development (EUD) [38] aims at putting customization mechanisms in the hands of end users, i.e., the subjects who are most familiar with the actual needs to be met [31]. Starting from iCAP [27], an early rule-based system for building context-aware applications, previous work demonstrated that EUD techniques can be effectively adopted to personalize the functionality of connected entities in different contexts, including mobile environments [42], smart homes [13, 52], and the web [24, 50]. Nowadays, in particular, several commercial platforms such as IFTTT [1], Zapier [3], and Microsoft Flow [2] are becoming popular [26]. These platforms typically adopt the Trigger-Action Programming (TAP) paradigm, i.e., they allow the definition of IF-THEN rules in the form of "*if* something happens, *then* perform an action." Users can therefore reuse or manually compose rules such as *"if I publish a photo on Facebook, then upload it to my Google Drive,"* or *"if the Nest security camera detects a movement, then blink the kitchen's Philips Hue lamp."* To *reuse* a rule, users can browse popular rules already created and shared by other people. To *define* a rule, instead, users have to link together a trigger and an action by exploiting wizard-based procedures, as shown in Figure 1.



(a) Rule composition in IFTTT



(b) Rule composition in Zapier

Fig. 1. Manualy composing a rule in platforms like IFTTT (a) and Zapier (b) means linking together a trigger and action. To define a trigger (or an action), users have to select a device or online service by searching in large menus of supported products. With the spread of new technologies, the amount of information may become too high, thus making the rule composition process difficult.

While TAP could potentially satisfy most of the behaviors desired by users [52], reusing and composing rules in contemporary TAP platforms is challenging. Previous work [20, 34, 52, 53] highlighted many interoperability, scalability, and understandability challenges that arise from the "low-level" of abstraction of the representation models adopted by such platforms. In these platforms, indeed, smart devices and online services are typically modeled on the basis of the

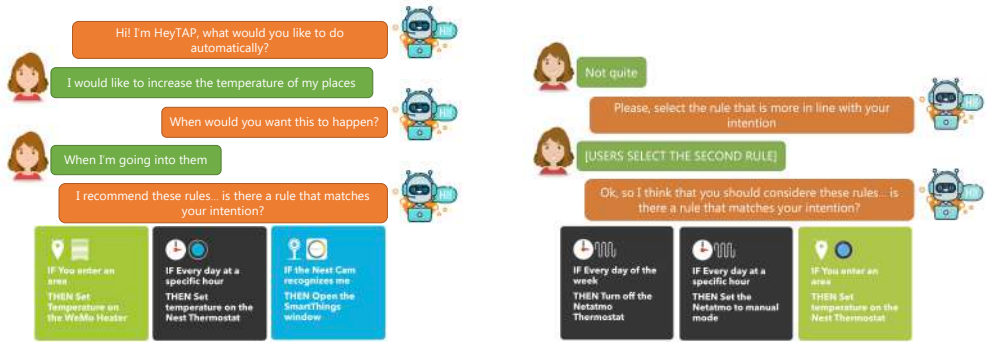
underlying brand or manufacturer [20], and as the number of supported technologies grows, so do the design space, i.e., the combinations between different triggers (*if*s) and actions (*then*s). Consequently, also the number of shared and publicly available rules is growing. This generates a problem of *information overload*: Zapier, for example, supports more than 1,000 devices and web applications, each one with its own triggers and actions, while the number of publicly available and reusable rules on IFTTT already exceeded 200,000 in September, 2016 [53]. Users are, moreover, forced to know in advance any involved technological detail, and they have to define several rules to program their ecosystems, i.e., every connected entity needs to be managed separately. As a result, they often experience difficulties in discovering rules and their related functionality [53], and TAP becomes a complex task for people without any previous programming experience [35].

Popular conversational agents such as Amazon Alexa [4] and Google Assistant [5] opened the way for interacting with smart devices and online services in a more abstract way, via conversation. Besides allowing users to ask for information and directly control other connected entities, some conversational agents can be exploited to set up *routines* in the form of trigger-action rules, e.g., to turn on a smart lamp when the user says a given sentence. Unfortunately, these end-user personalization capabilities are segregated in the mobile apps of these systems, only, and they take no advantages from the NLP and vocal capabilities of the devices on which they are installed [25]. Only few recent works [22, 35] started to investigate the possibility of using conversational systems to define IF-THEN rules. However, the proposed systems either adopt semi-automated procedures, e.g., by leveraging crowd workers [35], or they employ simplistic recommendation processes that greedily suggest *all* the rules that can be linked to the collected users' inputs [22].

In this paper, we present $HeyTAP^2$, a semantic Conversational Search and Recommendation (CSR) system able to *automatically* suggest pertinent IF-THEN rules that can be easily finalized and deployed in different contexts starting from an abstract description of a user's *personalization intention*. Differently from traditional recommendation systems, users can explicitly "guide" the recommendation process by specifying, in high-level terms, what they would like to personalize. Furthermore, unlike existing "single-shot" approaches, users can iteratively dialogue with $HeyTAP^2$ to dynamically refine the received recommendations, that become more precise as long as the interaction between the user and the system proceeds.

Through a natural language dialogue (exeplified in Figure 2a), the user first communicate her current high-level *intention* to a conversational agent, e.g., to increase the temperature of an indoor environment when she is going into it. The content of this conversation is mapped to information that can be extracted from reusable semantic vocabularies shared on the web [54]. Stemming from this extracted intention, $HeyTAP^2$ recommends n candidate IF-THEN rules that are both supported by the user's connected entities and that can satisfy the intention, while reflecting her long-term preferences revealed by her user profile. If the user cannot find a rule that fully satisfies her intention, $HeyTAP^2$ implements a *preference-based feedback* approach [49] by iteratively collaborating with the user to get further feedback and thus refining the recommendations. At each refinement cycle, in particular, the user can provide $HeyTAP^2$ with a short-term preference by selecting the recommended rule that is more in line with the intention (Figure 2b). Such an information is used to re-weight candidate rules and promote the recommendations of items that are similar to the provided feedback.

Besides presenting $HeyTAP^2$, we show how it can be configured to balance the current user's intention and her long-term preferences, as well as to take into account positive and negative feedback extracted during the refinement cycles between the user and the system. We evaluate such a CSR system through different offline experiments with simulated users and real-world data. First, we test the recommendation algorithm in different configurations, and we show that recommendation accuracy and similarity with target items increase as the interaction between the



(a) Intention Elicitation (b) Refinement Cycles

Fig. 2. $HeyTAP^2$ is a semantic-based Conversational Search and Recommendation (CSR) system that suggests IF-THEN rules for the personalization of smart devices and online services starting from an abstract user's need. The user can get recommendations by expressing her current personalization intention through a natural language dialogue with a conversational agent (a). If the user cannot find a rule that fully satisfies her needs, she can provide further feedback and get refined recommendations (b).

system and the user proceeds. Then, we compare $HeyTAP^2$ with recommended systems that are in the TAP domain or employ a *preference-based feedback*. Results are promising and demonstrate the effectiveness of $HeyTAP^2$ in recommending IF-THEN rules that satisfy the current personalization intention of a user.

Summarizing, the main contributions of our work are the following:

- We present $HeyTAP^2$, a CSR system that uses a semantic-based and conversational recommender algorithm to suggest pertinent IF-THEN rules for different contexts, based on the current personalization intention of the user and her long-term preferences, along with further feedback that can be extracted through natural-language dialogues. To the best of our knowledge, this is a novel approach that has never been explored before, especially in the TAP context.
- We evaluate $HeyTAP^2$ through different offline experiments involving simulated users and real-world data;
- We discuss the results of our work by highlighting the difference of our approach with previous attempts to simplify trigger-action programming, and we demonstrate that the usage of a conversational approach to get recommendations and refine them is a promising solution to "translate" an abstract user's need into executable IF-THEN rules. In particular, we show that $HeyTAP^2$ is able to provide recommendations that are more similar to the target item with respect to other recommendation systems in the trigger-action programming context, e.g., *RecRules* [21]. Furthermore, we demonstrate that, after some refinement cycles, $HeyTAP^2$ outperforms a state-of-the art*preference-based feedback* approach, i.e., *n-by-p* [44].

The remainder of this paper is structured as follow. Section 2 discusses related works. Section 3 presents the overall $HeyTAP^2$ system, by exemplifying its usage. Section 4 formally describes the conversational and semantic recommendation approach adopted by $HeyTAP^2$. The different offline experiments with simulated users and real-world data are reported in Section 5, while results and limitations are discussed in Section 6. Eventually, Section 7 concludes the paper.

2 RELATED WORK

2.1 Trigger-Action Programming in the Internet of Things: Issues and Opportunities

Thanks to the IoT, people are increasingly moving from passive consumers to active producers of information, data, and software [41]. The IoT, in particular, facilitates the creation of heterogeneous ecosystems [31] that can be exploited by users to access functionality and data provided by the so called "smart objects [10]," i.e., interconnected devices equipped with electronics, sensors, and actuators. In this scenario, even end users, i.e., the most suitable stakeholder to program the IoT [31], are willing to link together the different *behaviors* [26] exposed by smart devices, with the aim of accommodating their everyday needs [28]. Moreover, the wide adoption of online services such as social networks and messaging apps has further expanded the possibility of creating end-user applications in various domains [53].

Starting from iCAP [27], a visual rule-based system for building context-aware applications, the research community explored different EUD approaches in several contexts, including mobile environments [42], smart homes [13, 52], and the web [24, 50]. Therefore, it is not surprising that commercial programming platforms for personalizing connected entities such as IFTTT [1] and Zapier [3] are becoming popular [34]. The majority of such platforms [26] are based on trigger-action programming, one of the most popular paradigm to empower end users in directly programming their connected entities [27, 52]. By defining trigger-action rules, users can connect a pair of devices or online services in such a way that, when an event (the *trigger*) is detected on one of them, an *action* is automatically executed on the latter.

Despite defining IF-THEN rules may seem a simple task, recent studies have begun to take a closer look at this process, e.g., through empirical characterization of usage perfomances [40] and large-scale analysis of publicly shared rules [53]. Huang et al. [35] highlighted that users without programming skills may experience difficulties in dealing with TAP, while other researchers [34, 52, 53] criticized the expressiveness and understandability of platforms like IFTTT since they are rather limited. Barricelli and Valtolina [11], in particular, stated that contemporary TAP platforms often "*offers a too complex solution for supporting end users in expressing their preferences.*" Such a complexity can be associated with the *low-level of abstraction* of the adopted representation models [20], where every device and online service are treated as different entities, even if they provide the same "logical" functionality. If we consider the continuous growth of the IoT ecosystem, such an approach is clearly not scalable, and it generates a problem of information overload and seeking for the user interfaces of contemporary platforms [21]. Therefore, in presenting their empirical analysis of more than 200,000 IFTTT public rules, the largest-scale investigation of this type up to now, Ur et al. [53] highlighted that the explosion of entities and connections suggests the need to provide users with more support for discovering functionality and managing collections of IF-THEN rules. They emphasized, in particular, the need of making "IFTTT rules more expressive." Indeed, the representation models of TAP platforms also influence the ability of users to understand their IF-THEN rules. By systematically studying the impact of different IFTTT trigger and action types, Huang and Cakmak [34] found that users often misinterpret the behavior of trigger-action rules, often deviating from their actual semantics. Furthermore, a low-level of abstraction is often not aligned with users' mental models. According to Ur et al. [52], in fact, while the trigger-action approach can be both useful and usable for end-user development in IoT settings like smart homes, the level of abstraction end users employ to express triggers and actions needs to be better explored. In their user studies with more than 200 participants, for instance, the authors found that many users express triggers one level of abstraction higher, e.g., "when I am in the room" instead of "when motion is detected by the motion sensor." As Corno et al. [20] further confirmed in a controlled lab study with 24 participants, this suggests that, when defining trigger-action rules, the majority

of end users reasons about abstract personalization goals that can be addressed through multiple, similar IF-THEN rules at run-time, e.g., depending on the current contextual situation.

2.2 Overcoming the Low-Level of Abstraction Issue

Prior work tried to overcome the low-level of abstraction issue by exploring two approaches, mainly.

On the one hand, researchers started to explore the usage of *recommendation techniques*. Yao et al. [55] developed a probabilistic framework to suggest relevant smart objects to be personalized based on user interests. Besides using information about smart devices, the framework exploits social relationships between users to make more accurate recommendations. Corno et al. [21], instead, proposed RecRules, a hybrid and semantic recommendation system of IF-THEN rules that allows users to discover new rules on the basis of the devices' underlying functionality, rather than on the involved brands or manufacturers. The recommender integrates both content-based and collaborative information in a graph-based setting, and it extracts different types of features based on the underlying connections between graph's items to make top-N recommendations. The same algorithm has been integrated in a IFTTT-like platform [23], and it has been used to support the definition of IF-THEN rules, e.g., through auto-completion features. While advances in EUD have expanded the opportunities for offering recommendations [33], however, mapping IF-THEN rule suggestions to abstract and evolving users' needs is challenging, because suggestions often "*don't reflect exactly [users'] needs [23].*"

On the other hand, recent studies [20, 26, 31] tackled information overload in TAP platforms by envisioning the usage of a *higher level of abstraction*, with abstract IF-THEN rules that can be adapted to different contextual situations. Desolda et al. [26] defined a new representation for IF-THEN rules definition that combines multiple events and conditions exposed by smart objects, along with a three-layer architecture to support rules' execution. By separating interaction, logic, and service concepts, in particular, their architecture enables the definition of multiple front-ends addressing different execution platforms, i.e., different devices. Ghiani et al. [31], instead, proposed a generic model and its specialization in a home automation scenario to personalize the contextual behavior of IoT applications through trigger-action rules. Their environment is able to support the specification of flexible behaviors than can adapt to possible contextual changes. Similarly, Corno et al. [20] presented EUPont, a high-level representation for IoT personalization that allows users to define abstract and technology-independent IF-THEN rules like "if I enter a closed space, then illuminate it." Besides describing the model, the authors explored the advantages of adopting it to compose IF-THEN rules, showing that the usage of a higher level of abstraction allows users to reduce their errors and time effort. While using *fuzzy* triggers and actions [53] such as "when user is sleeping [31]" or "decrease the temperature in the room [20]" are in line with user's mental models [53], and could allow end users to personalize their connected entities without the need of explicitly programming every single involved device, a higher level of abstraction opens up new questions and challenges. How can a system decide how to "decrease the temperature" in a room? Is turning on the air conditioning system the right choice for the user? Does the user prefer to open the window, e.g., because she is interested in saving energy? In other words, while models like EUPont can be used to support the definition of abstract rules, then there is the problem of mapping abstract triggers and actions into executable commands over real devices and services.

In this paper, we take advantage of both the aforementioned approaches, i.e., recommendations and high-level of abstraction, by presenting a Conversational Search and Recommendation System able to traduce an abstract user's intention into pertinent IF-THEN rules that can be easily finalized and deployed in different contexts. Recently, Zhang et al. [57] proposed to automatically synthesize this kind of intentions into trigger-action programs by analyzing "users' traces", i.e., previous

manual interactions with smart devices. With $HeyTAP^2$, our aim is to directly involve users in the recommendation process by letting them specify their intentions. Consistently with the End-User Development vision [38], this may foster users' creativity and engagement, keeping users "in-the-loop" with their technologies. Furthermore, our approach is more generalizable, since it also allow users to personalize yet unknown connected entities.

While popular conversational agents such as Amazon Alexa [4] and Google Assistant [5] have opened the way for interacting with smart devices and online services via conversation, such a possibility is currently underexplored in the TAP context. Some recent studies in the smart-home context [56, 58] focused on the need of assisting end users to easily verify and check the run-time behavior of their defined IF-THEN rules via natural language, e.g., to avoid conflicts. Zhang et al. [56], for instance, presented a novel approach that can automatically generate formal LTL specifications for rules' verification from natural language requirements. With such an approach, users can check whether their IF-THEN rules respect high-level specifications like *"the cooler in the bedroom and the heater in the living room should not be on simultaneously."* Similarly, Zhang et al. [58] developed AutoTap, a system that allows users to specify, through a graphical user interface, a set of high-level properties that should always be satisfied by their devices. Such properties can be used to automatically repair conflicting rules, or to synthesize new rules from scratch. Rather than checking rules, our work aims at allowing end users to directly compose new IF-THEN rules via high-level conversation and recommendations. A first prototype to compose rules via conversation and recommendations, named InstructableCrowd, was proposed by Huang et al. [35]. InstructableCrowd is a crowd-sourcing system that enables users to create IF-THEN rules based on their needs. By exploiting a custom user interface on their smartphones, users can converse with crowd workers to describe some problems they are encountering, e.g., being late for a meeting. Crowd workers can therefore exploit a tailored interface to combine triggers and actions in recommended IF-THEN rules that are then sent back to the users' phones. In our work, our goal is to *automatically* map users' intentions to recommended IF-THEN rules, i.e., without the help of other users such as crowd workers. As stated by Funk et al. [29], indeed, we support the need of first capturing an end-user's intention and potential usage scenario, and then provide this information to a control system that learns how to effectively resolve such an intention in the specific user context. The work presented in this paper is built on top of our previous prototype to define IF-THEN rules via conversation [22]. Our previous tool allows end users to define their triggers and actions at different level of abstractions, and it uses a extremely simplistic "recommendation" process, i.e., it greedily shows the user *all* the rules that can be linked to the collected inputs. The number of suggested items is therefore not fixed, and users still have to browse a large number of rules to find a personalization that satisfies their needs. In this work, we stems from our prototype by adding the possibility of further filtering the proposed recommendations. To this end, we consider user's long-term preferences, and we better specify how to model abstract user's needs in the form of *personalization intentions*. Furthermore, we introduce a novel conversational recommendation process that is in charge of translating user's inputs into suggestions that can be iteratively refined.

2.3 Conversational Search and Recommendation Systems

Recommender systems were mainly invented to help users address information overload [37]. Through prediction models that take into account user's long-term preferences (e.g., ratings) and collaborative information (e.g., ratings of "similar" users), they recommend relevant items to the user by estimating how much she will like them. Recommendations are typically presented to the user through a single-shot approach, e.g., by ranking candidate items and displaying top-N suggestions. In the TAP context, the same RecRules algorithm recommends IF-THEN rules through a single-shot procedure [21]. Such an approach, however, is not particularly useful for suggesting

IF-THEN rules starting from a user's intention: users may not know in advance what is the best way to satisfy their goals, and personalization intentions may change over time by discovering what are the available triggers and actions in the specific context of use [22].

An alternative approach, commonly adopted in knowledge-based recommenders [15], is represented by *conversational* recommender systems [14, 47]. In conversational recommendation, users can provide feedback on the computed recommendations even without consuming them, with the aim of receiving back a refined set of suggestions. Through different cycles of interaction, the user can communicate her preferences to finally find a suitable item for her needs. Recently, the RecSys community started to investigate such a domain by referring to a new "conversational search and recommendation" research topic. Zhang et al. [59] stated that "conversational search" aims at finding or recommending the most relevant information for users based on textual or spoken dialogs. In our work, our "information" are IF-THEN rules: by interacting with the $HeyTAP^2$ conversational agent, the user can discover appropriate IF-THEN rules that can satisfy her abstract personalization intention as a result of a natural language dialogue with the system. Similarly, Jannach et al. [36] defined a Conversational Search and Recommendation (CSR) system as a software system that supports its users in achieving recommendation-related goals through a multi-turn dialogue. In our work, the goal of $HeyTAP^2$ is to recommend IF-THEN rules that satisfy the user's personalization intention. Through a multi-turn dialogue composed of the initial intention elicitation and the subsequent refinement cycles, $HeyTAP^2$ guides the user in discovering the functionality of her connected entities according to her preferences.

Previous work explored different strategies to elicit user's preferences during the recommendation process, ranging from simple selection-based feedback [44] to natural language dialogues [30]. In question-based recommendation systems like the *System Ask - User Respond* [59], for example, the system asks questions to the user about her preferred values for some modeled attributes. Similarly, a conversational recommendation systems can also ask the user to *criticize* recommendations [18], with the aim of understanding the attribute values that would improve the quality of the suggested items. There are domains, however, where answering these type of questions may be difficult. In the IoT context, for instance, expressing preferences at features or attributes level may be challenging for end users [20]. Under these circumstances, previous work suggests to adopt a *preference-based feedback* mechanism [49]. Similarly to relevance feedback techniques in information retrieval [45], the user can evaluate the received recommendations by expressing her *short-term preference* towards the current suggestions that are more in line with her current need. Such a mechanism simplifies the elicitation of users' preferences [48], and avoids the noise introduced by users when asked to explicitly rate items [6]. The recommendation process implemented in $HeyTAP^2$, in particular, is inspired by the *navigation-by-preference* (*n-by-p*) approach recently presented by Rana et al. [44] in the context of movies recommendations. Given a seed item, *n-by-p* helps the user navigate through the item space with recommendations that are computed by taking into account the user's history and her short-term preferences, i.e., the movies the user select during the different cycles of interaction with the conversational system. We use personalization intentions, elicited from the users via natural language dialogues, as a starting point to model user's short term preferences and "guide" the recommendation process. In addition, we support items enriched with semantic information, i.e., OWL classes[1], by taking advantage of the Semantic Web framework [8].

[1]https://www.w3.org/TR/owl2-overview/, last visited on May 11, 2020

2.4 Semantic Recommender Systems

As we exploit the Semantic Web framework to describe items, our work is also related to the body of research on semantic-based recommender systems. Several works on ontological recommender systems have been proposed in the literature [7, 16, 43, 46] with the aim of improving the performance of recommender systems, and to overcome cold start and data sparsity issues. The advent of initiatives like the Linked Open Data (LOD) [12] and the Linked Open Vocabularies (LOV) [54] opened the way for a new class of ontological recommender systems based on data freely available on the Web. In our work, we use information extracted from the EUPont ontology[2] [20] to model user's personalization intentions and finding similarities between IF-THEN rules. As it leverage semantic technologies, however, our approach is generalizable, and could potentially exploit different semantic representational models. Furthermore, it is worth noting that $HeyTAP^2$ is task-oriented, with a recommendation process that starts with an explicit user's input, i.e., the elicited personalization intention. This makes our recommender system less susceptible to cold-start problems than other approaches: when the underlying EUPont-based model is properly initialized, e.g., with the inclusion of the user's devices and services, the system is able to recommend IF-THEN rules even if the user has never defined any rule before. As such, the main limitation of the proposed approach lies in the maintainability of the underlying semantic model, that needs to dynamically reflect changes in users' contexts and owned connected entities. Implementing $HeyTAP^2$ in a new home, for instance, would require importing the EUPont ontology by instantiating the available smart devices. This is a known issue of semantic-based recommender systems that can be addressed in several ways, e.g., through frameworks for automatic ontology generation [51].

3 SYSTEM WALKTHROUGH

The $HeyTAP^2$ system consists of a web application, whose architecture is depicted in Figure 3. The *conversational agent* takes advantage of the Angular framework[3], while the *server*, which host the core parts of the CSR system described below, was developed with the Spring[4] framework.

$HeyTAP^2$ allows users to discover pertinent IF-THEN rules to satisfy their abstract personalization intentions with a two-phases procedure. First, it elicits the current intention of the user through a natural language dialogue, by taking advantage of a conversational agent (*personalization intention elicitation*). The $HeyTAP^2$ server mixes such an intention with the user context, i.e., information about her connected entities, and with the user profile, i.e., her long-term preferences, to recommend a first set of n IF-THEN rules. If the user is not fully satisfied with the received suggestions, she can iteratively collaborate with the agent to provide $HeyTAP^2$ with further feedback and get refined recommendations (*refinement cycles*). While the $HeyTAP^2$ system can be generalized to work with IF-THEN rules of different TAP platforms, we implemented it to support all the triggers and actions of IFTTT. We made this choice by considering the popularity of the platform [34] and the availability of open datasets [53].

3.1 Personalization Intention Elicitation

To specify a personalization intention, the user collaborates with the $HeyTAP^2$ conversational agent (Figure 3, a). A personalization intention is an abstract description of a behavior (the *"what"*) that the user would like to be automatically executed in a specific context (the *"when"*), e.g., to increase the temperature of an indoor environment when she is going into it. The agent exploits the

[2]http://elite.polito.it/ontologies/eupont.owl, last visited on May 15, 2020
[3]https://angular.io/, last visited on April 26, 2020
[4]https://spring.io/, last visited on April 26, 2020

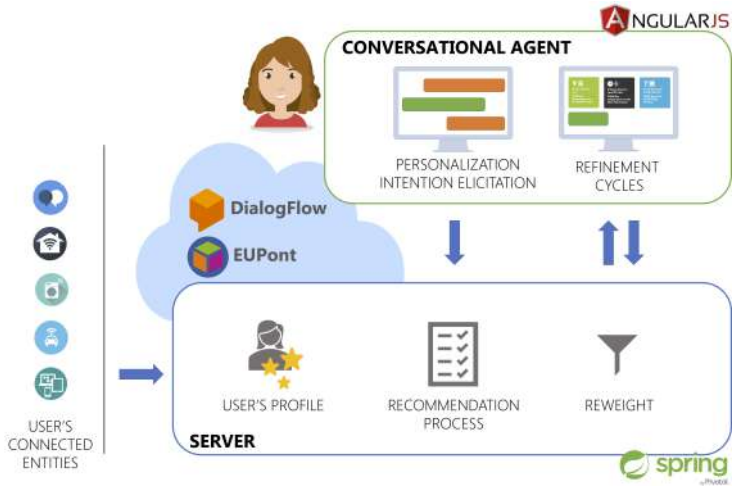
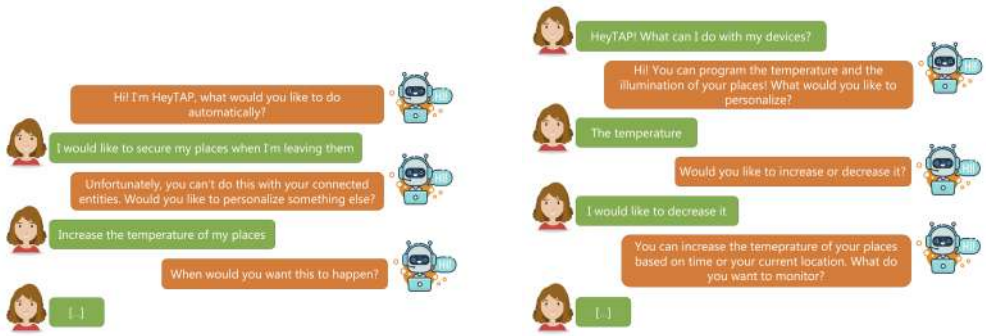


Fig. 3. The architecture of the $HeyTAP^2$ system. The user intention elicited from the user through the conversational agent is mixed with information about the user's connected entities and the user's profile, i.e., the long-term preferences, to rank candidate rules on the basis of their semantic information, previously extracted from the EUPont ontology. Any additional feedback provided by the user during the refinement cycles is used to re-weight candidate rules.

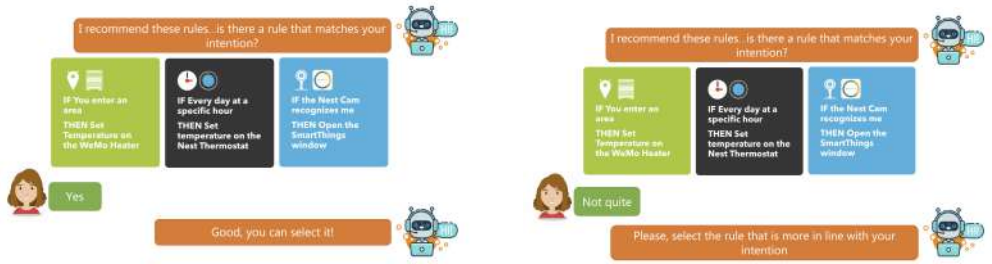


(a) User-Inititative Elicitation (b) System-Inititiative Elicitation

Fig. 4. The two strategies supported by $HeyTAP^2$ to elicit personalization intentions. With the user-initiative elicitation (a), the user can directly specify her evisioned intention. With the system-initiative elicitation (b), instead, $HeyTAP^2$ "guides" the user in discovering what she can personalize with her connected entities.

DialogFlow[5] framework to interpret user inputs. If the input is recognized by DialogFlow, it is sent to the $HeyTAP^2$ server. The server analyzes the user's input and generates the responses for the user, e.g., to ask for more details, or to notify the user about unrecognized behaviors. When both the *"what"* and the *"when"* of the intention have been successfully collected, the server maps them onto OWL classes extracted from the EUPont ontology [20]. Then, it combines such an information with long-term preferences extracted from the user's profile, by ranking all the candidate rules of the user recommendations set, i.e., all the rules yet unknown to the user but that are supported by

[5]https://dialogflow.com/ last visited on April 26, 2020



(a) Consume a recommendation (b) Start a refinement cycle

Fig. 5. When the user receives a set of n recommended rules by $HeyTAP^2$, she can select a rule to be used (a), or she can provide $HeyTAP^2$ with further feedback to refine recommendations (b). As an example, the figures report $HeyTAP^2$ recommending $n = 3$ IF-THEN rules at a time.

her owned connected entities. The ranked list is finally used to provide the user with a first set of n recommendations (see Section 4 for details on this recommendation process).

As shown in Figure 4, $HeyTAP^2$ supports two different strategies to elicit users' intentions. In the *user-initiative elicitation* (Figure 4a), the user can directly express her personalization intention by specifying her envisioned "what" and "when". When the $HeyTAP^2$ server is not able to map the user's input onto supported concepts or user's context, i.e., her owned connected entities (see Section 4.1.1 for further details), it notifies the user through the conversational agent, by allowing her to specify a different or refined intention.

Alternatively, $HeyTAP^2$ can elicit users' intentions through a *system-initiative procedure* (Figure 4b), during which the system"guides" the user in discovering the "what" and "when" supported by her connected entities. This is needed bacause, as demonstrated by the previous study on our prototype to define IF-THEN rules via conversation [22], users are often not aware of the functionality offered by their devices and services, and they can therefore experience difficulties in thinking about possible personalizations without any help.

Independently of the adopted strategy, the $HeyTAP^2$ server uses the collected intention to perform a semantic ranking of supported IF-THEN rules and recommend n of them.

3.2 Refinement Cycles

When the user receives a set of recommended rules by $HeyTAP^2$, she has two possibilities. If she can find a rule that matches her personalization intention, she can terminates the recommendation process by consuming it (Figure 5a). Conversely, if the user is not fully satisfied with the received suggestions, she can start a refinement cycle (Figure 5b). $HeyTAP^2$ asks the user to select the recommended rule that is more in line with her current intention, i.e., a *short-term preference*. The $HeyTAP^2$ server, then, uses such a further feedback to re-weight the supported IF-THEN rules and recompute recommendations (see Section 4.3 for further details on the re-weighting process).

3.3 Running Example

Let us consider a user whose intention is to personalize the temperature of her indoor places when she is going to enter them. By exploiting contemporary programming platforms like IFTTT and Zapier, she is forced to *manually* compose her trigger-action rules by linking together different triggers and actions (see Figure 1, for example). This implies that the user must know *in advance* which connected entities, triggers, and actions to use to achieve her intention. For instance, she

should be aware that her *Nest* security camera provides a trigger to detect the fact that she is entering home. As previous works already highlighted [20, 53], however, the majority of end users do not have this knowledge, and the large menus of supported entities, triggers, and actions offered by contemporary platforms are often browsed without any success [20].

(a) (b)

Fig. 6. The user communicates a temperature-related personalization intention to the conversational agent of $HeyTAP^2$ (a). The system uses the intention and the user's long-term preferences to produce a first set of recommendations (b).

Figure 6, Figure 7, and Figure 8 show all the steps that the user can follow to translate her temperature-related intention into executable IF-THEN rules through $HeyTAP^2$. By exploiting the conversational agent built as an Angular web application, the user can directly communicate the intention to $HeyTAP^2$ via natural language (Figure 6a). Differently from contemporary programming platforms, $HeyTAP^2$ assumes that the user has a limited knowledge of the functionality offered by her connected entities. Rather, its goal is to *guide* the user in discovering appropriate IF-THEN rules that can satisfy her abstract need of personalizing the temperature of her indoor places. $HeyTAP^2$, in particular, mixes the collected intention with the rules that the user has already defined in the past, i.e., the user's long-term preferences, and suggests the 3 rules of Figure 6b, i.e.:

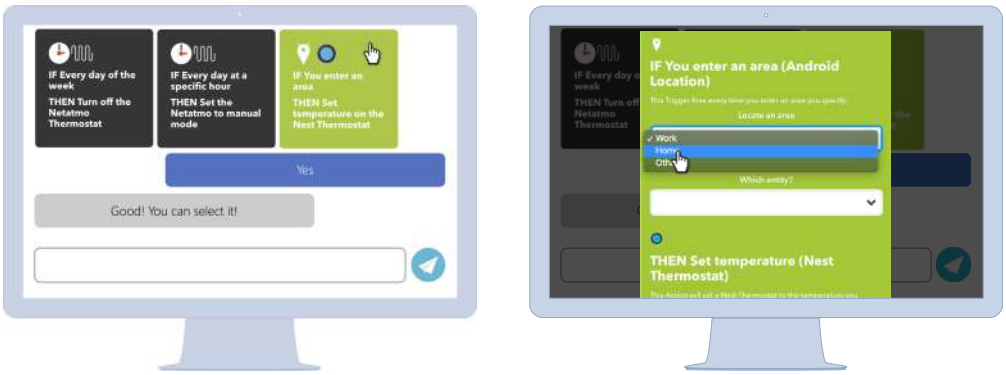
R1 **IF** the *Android* smartphone detects that you entering a geographical, **THEN** set the temperature on the *WeMo* heater;
R2 **IF** every day at a specific hour (*Date & Time* service), **THEN** set the temperature on the *Nest* thermostat;
R3 **IF** the *Nest Cam* recognizes me, **THEN** open the *SmartThings* window.

Unfortunately, the user cannot find a satisfying rule in the first 3 recommendations. By interacting with the conversational agent (Figure 7a), however, she selects **R2** as the rule that is more in line with her intention. The aim of this phase is to allow $HeyTAP^2$ to further refine recommendations and make the user discover the right rule for her need. As shown in Figure 7b, the system exploits the user's short-term preference to recommend 3 alternative rules:

R4 **IF** every day at a specific hour (*Date & Time* service), **THEN** turn off the *Netatmo* thermostat;
R5 **IF** every day at a specific hour (*Date & Time* service), **THEN** set the *Netatmo* thermostat to manual mode;
R6 **IF** the *Android* smartphone detects that you entering a geographical, **THEN** set the temperature on the *Nest* thermostat.

(a) (b)

Fig. 7. The user is not satisfied with the current suggestions, but provides a short-term preference feedback to "guide" the recommendation process of $HeyTAP^2$ (a). The system uses the short-term preference to refine recommendations (b).



(a) (b)

Fig. 8. After the refinement cycle, the user finds a rule that satisfies her personalization intention (a). The user activates the rule by specifying the required details (b).

Intuitively, $HeyTAP^2$ polarizes the recommendations around the connected entities involved in **R2**, i.e., the *Date & Time* service and the *Nest* thermostat. Now, the user is satisfied with the proposed suggestions: as shown in Figure 8a, she selects **R6**. The conversational agent makes her complete the rule with the required details (Figure 8b), and activates the rule on the involved connected entities.

4 $HeyTAP^2$ RECOMMENDATION PROCESS

To provide users with recommendations and to iteratively refine them, $HeyTAP^2$ implements the semantic and conversational recommendation process outlined in the previous section. In this section, we formally describe it.

Let T, A, and R be the sets of all the triggers, actions, and rules supported by the connected entities owned by the user, with $R = \{(t, a) : t \in T, a \in A\}$. The recommendation process works in

a scenario of implicit ratings, where the user profile $P \subseteq R$ is a set of rules with which the user has interacted in the past. Such an interaction can be defined in different ways, e.g., by taking into account the creation of a rule, or the reuse of a rule shared by another user. Given the user profile P, the recommendation set RS, i.e., the rules that might be recommended to the user, includes the supported rules that do not belong to P, i.e., $RS = \{r : r \in R \setminus P\}$.

Algorithm 1 $HeyTAP^2$ recommendation process

Input:
I: a user intention
n: number of final recommendations
Output:
k: a rule k to satisfy the intention I

$IC \leftarrow INTENTION\_CANDIDATES\ (I)$
if $|IC| = 0$ **then**
  return unsupported intention
$L \leftarrow PREFERENCES\ (P)$
$history \leftarrow []$
while $|RS| > n$ **do**
  $rec \leftarrow RECOMMEND\ (IC, L, n)$
  $s, k \leftarrow$ user chooses $s \in \{SATISFIED, NOTSATISFIED\}$ and $k \in rec$
  if $s = SATISFIED$ **then**
    return k
  $REWEIGHT(k, rec, \Phi)$
  $history \leftarrow history \cup R$
  $RS \leftarrow IC \setminus history$

The recommendation process is described in Algorithm 1, while its Java implementation is available on the web at https://git.elite.polito.it/public-projects/intrec. To get started, the process analyzes the user's intention I elicited through DialogFlow. Such an intention contains a description, in natural language, of "what" the user would like to personalize, and "when." By exploiting a common vocabulary that associates the DialogFlow inputs to semantic information, the process tries to map I onto OWL classes extracted from the EUPont ontology [20]. The output of this phase is a set IC of *intention candidates rules*, i.e., rules that belong to the user's recommendation set RS and that can be semantically associated to the user's intention.

If the set IC is empty, the recommendation process ends, and the $HeyTAP^2$ conversational agent notifies the user that her intention is not supported. It is worth noting that the fact that the set of intention candidate rules is empty does not reflect a cold-start problem, but rather:

- a problem with the initial input, e.g., when the user's sentences are not recognized, or,
- a problem with the underlying EUPont-based model, e.g., when it is not up to date and it does not include the full range of connected entities that can be controlled by the user.

If the set IC is not empty, the recommendation process continues. Note that the generation of the set IC is only a preliminary step of the recommendation process, and intention candidate rules do not represent the final recommendations submitted to the user. In order to produce a set of final recommendations, $HeyTAP^2$ iterates over the generated intention candidate rules by applying several criteria. In particular, $HeyTAP^2$ analyzes the user's profile P to extract the user's long-term preferences L, and computes a first set of n recommendations rec that depends both on L and IC. If the user has never defined any rules, i.e., L is not available, $HeyTAP^2$ is still able to perform recommendations by focusing on IC, only.

The computation of recommendations is semantic-based as well, i.e., it is performed by modeling IF-THEN rules on the basis of semantic information extracted from EUPont. If the user is satisfied with the received suggestions, the recommendations process ends. If not, recommendations can be iteratively refined until there are sufficient items in the recommendations set. Refining recommendations means re-weighting the rules in the recommendation set *RS* and recomputing recommendations. The re-weighting procedure applies a given policy Φ that takes into account the short-term preference k indicated by the user. As in *n-by-p* [44], we decided to not to recommend an item more than once during the same recommendation process. We therefore keep track of the *history* of the computed recommendations, by excluding them from the recommendation set.

In the following, we describe the details of each part of the recommendation process, ranging from the semantic modeling of intentions and rules to the re-weighting procedure.

4.1 Semantic Modeling of Personalization Intentions and Rules

The *HeyTAP*[2] recommendation process exploits semantic information available on the web for *a*) modeling the personalization intentions that the user can communicate via conversation, and *b*) modeling IF-THEN rules and recommend them. To this end, we take advantage of the EUPont ontology, and, in particular, its instantiation for IFTTT[6]. The model categorizes each trigger and action of the popular platform through a set of OWL classes $c \in C$ modeling their category and provided functionality. As shown in Figure 9 and Figure 10, functionality classes belong to different categories, e.g., `Temperature` and `Location`, and are organized hierarchically in two levels:

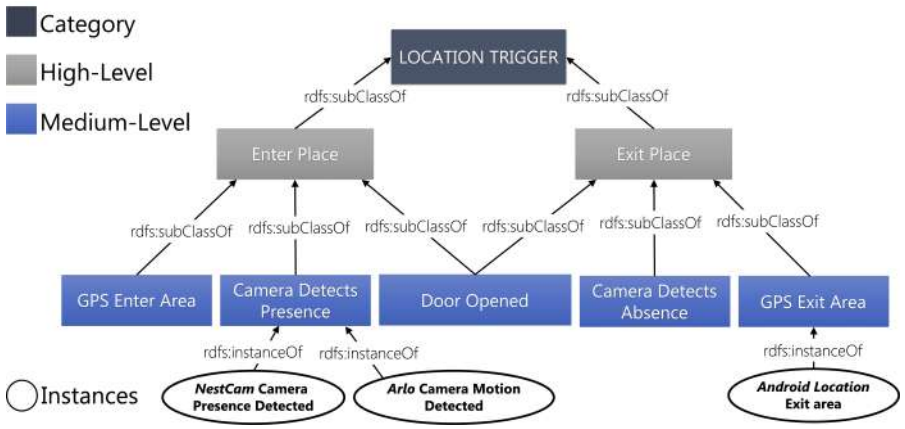


Fig. 9. A partial view of the hierarchical class tree that characterizes location-related triggers.

- High-Level (*HL*) classes model generic event to be verified, e.g., `Exit-Place`, or actions to be executed, e.g., `Lower-Temperature`, and they do not include any technical details, nor the type of device or service to be used to implement the desired behavior.
- Medium-Level (*ML*) classes model specific events to be verified, e.g., `Door-Opened`, or actions to be executed, e.g., `Set-Thermostat-Temperature`, by referring to the generic type of device or service that is involved.

IFTTT triggers and actions, e.g., "*NestCam* Camera Presence Detected" and "Set *Nest* Thermostat Temperature", are modeled as instances of *ML* classes.

[6]http://elite.polito.it/ontologies/eupont-ifttt.owl, last visited on May 15, 2020

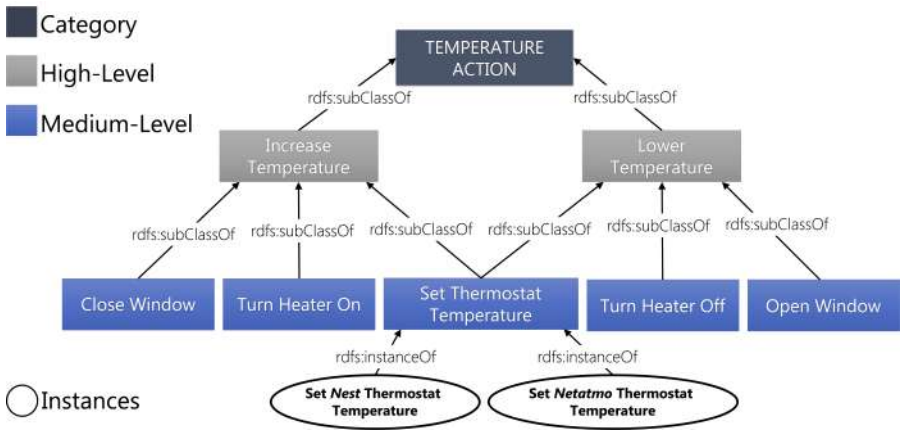


Fig. 10. A partial view of the hierarchical class tree that characterizes temperature-related actions.

4.1.1 Personalization Intentions Mapping. As reported in Section 3.1, a personalization intention is an abstract description of a behavior (the *"what"*) that the user would like to be automatically executed in a specific context (the *"when"*), e.g., to to increase the temperature of an indoor environment when she is going into it. More formally, $HeyTAP^2$ tries to map each personalization intention I over a set of EUPont classes $C_a \cup C_t$, where:

- C_a is a set of EUPont classes describing *"what"* the user would like to personalize;
- C_t is a set of EUPont classes describing *"when"* she envision the personalization.

Such a mapping is automatically performed by DialogFlow, that associates the recognized inputs and the OWL classes included in the EUPont ontology through a common vocabulary. The vocabulary with which we trained DialogFlow was extracted from our formative study on our prototype to define IF-THEN rules via conversation [22]. Both C_t and C_a are composed of two EUPont classes: an EUPont category and an EUPont HL functionality. This choice is motivated both from the literature, which highlights the benefits of using "fuzzy" triggers and actions in line with user's mental models (e.g., [52, 53]), and by the results of our formative study [22]. During the study, indeed, we found that participants rarely referred to specific devices and online services, but they often specified personalizations in an abstract way, by mentioning generic categories and functionality. Table 1 reports some examples of personalization intentions supported by $HeyTAP^2$. Users can refer to their "physical" word (e.g., the temperature of an environment) and their "virtual" word as well (e.g., notifications and news). Each intention includes a category and a functionality describing an action to be executed, and a category and a functionality describing a trigger to be monitored. Such information, in particular, are mapped onto OWL classes included in the EUPont ontology.

After mapping an intention I on EUPont classes, $HeyTAP^2$ extracts a set IC of *intention candidates* rules through the function reported in Algorithm 2. Through the *GET_INSTANCES* method, the function extracts all the triggers and actions categorized under the EUPont classes included in the intention I. Then, it combines them to generate the complete set of intention candidate rules IC. IC is therefore defined as the set of rules that can be generated from the OWL classes included in the intention I, i.e., C_t and C_a:

$$IC = \{(t, a) : C_t \subseteq Types\,(t)\,, C_a \subseteq Types\,(a)\}\,, \tag{1}$$

	Action Category	Action Functionality	Trigger Category	Trigger Functionality
Decrease the temperature of an environment when I'm leaving it	Temperature	Decrease-Temperature	Location	Exit-Place
Illuminate a place when I am arriving in it	Lighting	Increase-Lighting	Location	Enter Place
Secure a place when I am leaving it	Security	Increase-Security	Location	Exit-Place
Increase the temperature of an environment when the temperature decreases	Temperature	Increase-Temperature	Temperature	Decreased-Temperature
Send me a notification when there are available news	Information	Send	Information	News-Available
Send me a notification when the air quality of my environments decreases	Information	Send	Air-Quality	Decreased-Air-Quality

Table 1. Some examples of personalization intentions modeled in $HeyTAP^2$. Each intention includes a category and a functionality describing an action to be executed, and a category and a functionality describing a trigger to be monitored. Such information refer to available EUPont OWL classes. Intentions can refer to the physical word (e.g., the temperature of an environment) and the virtual word as well (e.g., notifications and news).

where $Types(x)$ is a recursive function that returns all the OWL classes (including super-classes) of the individual x.

Algorithm 2 Intention candidates extraction

Input:
I: a user intention
Output:
IC: a set of intention candidates rules

```
1: function INTENTION_CANDIDATES(I)
2:     T, A, IC ← []
3:     for c_t in I.C_t ⊂ I do
4:         T ← T ∪ GET_INSTANCES(c_t)
5:     for c_a in I.C_a ⊂ I do
6:         A ← A ∪ GET_INSTANCES(c_a)
7:     for t in T do
8:         for a in A do
9:             IC ← IC ∪ {(t, a)}
```

To further exemplify the personalization intentions mapping, let us consider the last user's personalization intention of Table 1, i.e., *"send me a notification when the air quality of my environments decreases."* Figure 11 illustrates how $HeyTAP^2$ relates the intention to actual triggers and actions. When the conversational agent receives the user's input, it uses the DialogFlow framework to recognize the trigger and the action part of the intention, as well as a category and a functionality entity for each part. In Figure 11, categories are highlighted in green, while functionality are

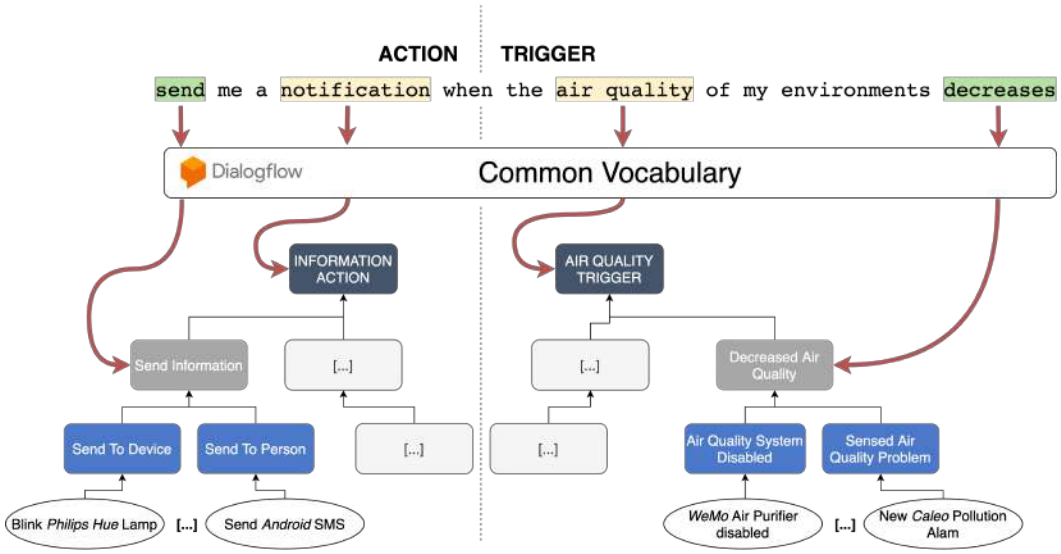


Fig. 11. The figure exemplifies how $HeyTAP^2$ maps the intention *"send me a notification when the air quality of my environments decreases"* to actual triggers, e.g., "New *Caleo* Pollution Alarm," and actions, e.g., "Send *Android* SMS."

highlighted in yellow. Thanks to the common vocabulary, $HeyTAP^2$ is able to map the extracted entities to EUPont classes: *"notification"*, for instance, is linked to the `Information-Action` class, while *"decreases"* is mapped to the `Decreased-Air-Quality` class. As shown in the bottom part of Figure 11, $HeyTAP^2$ uses such a mapping to "point" to the involved classes in the corresponding EUPont hierarchical class trees. By descending the identified branches of these trees, in particular, $HeyTAP^2$ is able to extract triggers, e.g., "New *Caleo* Pollution Alarm," and actions, e.g., "Send *Android* SMS", that can satisfy the user's intention. Extracted triggers and actions are finally mixed together to generate intention candidate rules.

4.1.2 IF-THEN Rules Features and Similarities. Each rule i is described through a set of features denoted f_i. Most recommendation approaches describe items in terms of textual features, e.g., keywords or tags. The main drawback of this kind of representation is that it completely ignores the *semantics* underlying the item space [43]. The semantic of a rule plays a fundamental role in $HeyTAP^2$, since the goal of the recommendation process is to allow users to discover the right rules to satisfy their personalization intentions, independently of the brands or manufacturers of the involved connected entities. For this reason, the set of features f_i describes the rule i according to its *meaning*, and it is extracted through a semantic reasoning process over the EUPont ontology. Given a rule i, in particular, its feature set f_i includes all the OWL classes $c \in C$ of the trigger and the action of i, i.e., $f_i = Type\,(t_i) \cup Type\,(a_i)$. As shown in Figure 9 and Figure 10, EUPont classes are organized hierarchically. Super-classes define generic concepts that can be further described by means of more specific nested classes.

With the described features, we can calculate the *semantic similarity* $sim\,(i, j)$ between two rules i and j, by exploiting the Jaccard similarity index of their semantic features:

$$sim\,(i, j) = \frac{|f_i \cap f_j|}{|f_i \cup f_j|} \tag{2}$$

Semantic similarities vary according to the position of the involved triggers and actions in the hierarchy of EUPont classes. The similarity of two rules that have both the trigger and the action categorized under the same EUPont ML classes is high. As shown in Figure 9, for example, the two IFTTT triggers "*NestCam* Camera Presence Detected" and "*Arlo* Camera Motion Detected," share different OWL classes, i.e., `Camera-Detects-Presence`, `Enter-Place`, and `Location`. Semantic similarity decreases when triggers or actions are categorized under different EUPont classes: in Figure 9, the trigger "*Android* Location Exit area" only shares a category class (`Location`) with the other IFTTT triggers shown in the hierarchy.

4.1.3 Long-Term Preferences Extraction. $HeyTAP^2$ exploits semantic similarities to extract the user's long-term preferences L as well. User's long-term preferences, in particular, are represented by candidate items in RS that are "semantic peers" of the rules in the user's profile P:

$$L = \cup_{i \in P} S_i \tag{3}$$

The "semantic peers" concept is similar to the concept of "neighbors" [44]: S_i are the candidate item in the recommendation set RS whose semantic similarity to i (Equation 2) exceeds a threshold θ:

$$S_i = \{r \in RS, r \neq i : sim\,(i, r) > \theta\} \tag{4}$$

Qualitatively, L contains a set of rules that could be potentially recommended (i.e., that belong to RS) and that are similar to the rules that the user has already defined in the past (i.e., that belong to P). Past user's interaction is therefore mapped to the user's recommendation set, and it is used to extract candidate rules that reflect the user's long-term preferences up to a specific moment.

4.2 Computing Recommendations

Given a personalization set of intention candidate rules IC and the user's long-term preferences L, we define the recommendation problem as the selection of n rules from the recommendation set RS that:

a) are similar to the rules in IC, i.e., they can satisfy the current user's intention, and,
b) reflect the user's long-term preferences L, i.e., what the user has done before with her connected entities.

Algorithm 3 $HeyTAP^2$ recommendation algorithm

Input:
IC: the set of intention candidate rules
L: candidate rules that are semantic pairs of rules in P
n: number of final recommendations
Output:
rec: a list of n recommended rules

```
1: function RECOMMEND(I, L, n)
2:     rec ← []
3:     while |rec| < n and |RS| > 0 do
4:         i_best = arg max_{i∈RS} score (i, L, IC)
5:         rec ← rec ∪ {i_best}
6:         RS ← RS \ {i_best}
       return rec
```

The recommendation algorithm (Algorithm 3), in particular, recommends the set rec composed of the n rules of RS that have the highest *score*. The formula to calculate the score of rule i takes

into account both IC, i.e., the current intention, and L, i.e., the long-term preferences of the user:

$$score\,(i, I, L) = (1 - \eta) \cdot sa\,(i, IC) + \eta \cdot sa\,(i, L \setminus IC) \tag{5}$$

The parameter η in $[0, 1]$ controls the balance between current intention and long-term preferences. To avoid counting the same rule twice, we exclude the rules that are also intention candidates from the set L. The factor $sa(i, X)$ measures the *semantic association* between i and a set of items X, where X is either IC or $L \setminus IC$:

$$sa(i, X) = \frac{\sum_{j \in X} sim\,(i, j) \cdot w_j}{\sum_{j \in X} w_j} \tag{6}$$

As shown in the formula, $sa(i, X)$ is the weighted average of the semantic similarities between the item i and the items in X. Each item $i \in RS$ has a weight w_i that is initially set to 1. Weights can be recomputed in the refinement cycles to reflect the further feedback that the user can provide to $HeyTAP^2$. Overall, Equation 5 clearly highlights why $HeyTAP^2$ is less susceptible to cold-start problems than other approaches. Indeed, when long-term preferences are not available, i.e., $|L| = 0$, the score of a rule can still be computed by considering the first addendum, only, i.e., by exclusively leveraging the semantic association of the rule with the set IC of intention candidate rules.

4.3 Recommendations Refinement

To refine recommendations, $HeyTAP^2$ uses the user's short-term preference k, i.e., the rule in rec that is more in line with the user's intention, to recompute the weights of the the rules in RS. Table 2 reports the re-weighting policies Φ supported by $HeyTAP^2$. Depending on the adopted policy Φ, $HeyTAP^2$ can take into account positive feedback, i.e., the item k, negative feedback, i.e., the items not selected by the user ($rec \setminus \{k\}$), or both of them.

Policy	Description
Φ_+	Promote items that are similar to the user's feedback k, do not penalize items.
Φ_-	Penalize items that are similar to the discarded items $rec \setminus \{k\}$, do not promote items.
$\Phi_\pm$	Promote items that are similar to the user's feedback k, penalize items that are similar to the discarded items $rec \setminus \{k\}$

Table 2. Re-weighting policies supported by $HeyTAP^2$

As reported in Algorithm 4, given an item $i \in RS$, $HeyTAP^2$ computes a) the semantic similarity of i with the rule k, i.e., sim_s, and b) the average semantic similarity of i with the discarded items $rec \setminus \{k\}$, i.e.,sim_d. Depending on these two values, the item i can be *promoted* or *penalized* by acting on its weight w_i. To promote and penalize items, in particular, we exploit the *Smean* function presented in [44], that consists in adding (or subtracting) a given quantity to an item's weight according to the current user's short-term feedback. More specifically, the function

- promotes an item that is similar to the short-term feedback k by incrementing its weight with the similarity between the item and k;
- penalize an item that is similar to the discarded items by decrementing its weight with the mean similarity between the item and the discarded items.

We exploited the *Smean* function since previous work [44] already demonstrated its superiority over other re-weighting policies in preference-based feedback approaches. In our work, in particular:

- An item is promoted when the re-weighting policy is Φ_+ or $\Phi_\pm$, and when the sim_s value is greater than the sim_d value, i.e., when the item is more similar to the rule selected by the user (the short-term preference k) than to the rules that have not been selected (the discarded items). In this case, the corresponding weight is updated as follow: $w_i = w_i + sim_s$.
- An item is penalized when the re-weighting policy is Φ_- or $\Phi_\pm$, and when the sim_d value is greater than the sim_s value, i.e., when the item is more similar to the rules that have not been selected (the discarded items) than to the rule selected by the user (the short-term preference k). In this case, the corresponding weight is updated as follow: $w_i = w_i - sim_d$.

Algorithm 4 *HeyTAP*^{2} re-weighting procedure

Input:
rec: the set of current recommendations
k: the user's short-term preference
Φ: the re-weighting policy

function REWEIGHT(rec, k, Φ)
  for i in RS **do**
    $sim_s \leftarrow sim\,(i, k)$
    $sim_d \leftarrow \sum_{j \in rec \setminus \{k\}} sim\,(i, j)\,/|rec \setminus k|$
    if $sim_s > sim_d$ and $(\Phi_+$ or $\Phi_\pm)$ **then**
      $w_i \leftarrow w_i + sim_s$
    if $sim_d > sim_s$ and $(\Phi_-$ or $\Phi_\pm)$ **then**
      $w_i \leftarrow w_i - sim_d$

5 EVALUATION

We evaluate the *HeyTAP*^{2} recommendation process through different offline experiments with simulated users and real-world data. To this end, we use a dataset of IF-THEN rules extracted from IFTTT [53] to simulate the interaction between users and *HeyTAP*^{2} by implementing established procedures described in previous works [32, 44] (see Section 5.2.1). We opt for offline experiments to quantitatively investigate the following research questions:

RQ1: assessing the recommendation process of *HeyTAP*^{2} in different configurations (e.g., by varying its parameters and the re-weighting policy);

RQ2: comparing the recommendation process of *HeyTAP*^{2} with other state-of-the-art recommeder systems.

By exploring the first research question, i.e., the assessment of the recommendation process in different configurations, we highlight the benefits of taking into account both users' intentions and long-term preferences. We show, in particular, that recommendation accuracy and similarity with target items increase as the interaction between the system and the user proceeds.

Through the second research question, instead, we compare *HeyTAP*^{2} with baseline recommender systems in the TAP context and exploiting *preference-based feedback* approaches. We investigate, in particular, the potential of our approach, and we describe its differences with other solutions.

5.1 Exploited Dataset and Metrics

5.1.1 Dataset. To the best of our knowledge, the dataset of IFTTT rules collected by Ur et al. [53] is the *only* publicly available dataset of IF-THEN rules defined and shared by different users. The included triggers and actions, in particular, are referenced by a specific instantiation of the EUPont

ontology [19]. The original dataset was obtained by Ur et al. [53] with a web scrape of the IFTTT platform as of September 2016. It contains 295,156 rules created and shared by 129,206 different authors, and has a high degree of sparsity (97.51%, Table 3).

Table 3. IFTTT Dataset statistics.

Metric	Value
Users (#)	129,206
Items (#)	295,156
Sparsity (%)	97.51

Each item of the dataset describes the creation of rule in the IFTTT platform: it includes information about the trigger, the action, and the author that created the rule. We used the rules created by a given user as the implicit feedback to build her user's profile P. We preprocessed the dataset by removing the users with less than 5 created rules. At the end, the dataset was composed by 163 users and 10,626 implicit feedback, overall. Since the dataset does not contain any information about which device or online service can be actually controlled by each user, we suppose that each user is authorized to control *any* connected entity, i.e., the recommendation set RS of a user potentially includes all the rules included in the dataset.

5.1.2 Metrics. Differently from recommender systems in other domains, e.g., movies, $HeyTAP^2$ is task-oriented, i.e., its goal is to allow users to discover a specific IF-THEN rule that that can satisfy the current user's intention. To investigate the task-oriented nature of $HeyTAP^2$, we build the test set by randomly picking a single *target* rule from each user's profile, and then we use the following two metrics averaged over all users:

- **HIT@N-C:** the hit-rate up to a given cycle c, i.e., the proportion of users who have been recommended their target item in the n recommendations after c refinement cycles;
- **SIM@N-C:** the Jaccard similarity of the n recommended rules in the current refinement cycle c with the target item.

Besides assessing the capability of the algorithm to recommend *exactly* the target item, we also measure, in terms of precision and recall, the ability of $HeyTAP^2$ of suggesting rules belonging to the set IC of intention candidate rules, i.e., rules that can satisfy the initial user's intention:

- **PREC$_{ic}$@N-C:** the fraction of the the recommended rules that are relevant to the current user's intention, i.e., that belong to the set IC, among all the rules that have been recommended up to the refinement cycle c;
- **REC$_{ic}$@N-C:** the fraction of the the recommended rules that are relevant to the current user's intention, i.e., that belong to the set IC, over the total amount of rules that are relevant, i.e., the IC size, up to the refinement cycle c.

Since recommendations are expected to be consumed in a conversational agent, and given the task-oriented nature of the recommendation process, the number n of computed recommendations needs to be limited. For this reason, in our experiments, we set $n = 3$.

5.2 Recommendation Process Assessment

To quantitatively assess the recommendation process of $HeyTAP^2$ (**RQ1**), we test it in different configurations, by varying the parameters of the algorithm, as well as the adopted re-weighting policy. Through such an assessment, we investigate how the recommendation accuracy and similarity with target items evolve as the interaction between the system and the user proceeds.

5.2.1 Experimental Setting. To simulate the interaction between a given user and $HeyTAP^2$, we proceed as follow. First, we simulate the user's intention by extracting the EUPont classes C_a and C_t for the user's target item. Stemming from the extracted classes, we compute the set of intention candidates rules IC. Furthermore, we use the user's profile P (excluding the target item) to compute the user's long term preferences L. A possibility to simulate the user's short term preference k in each refinement cycle is to choose it randomly from the current set of recommendations. As highlighted in [44], however, such an approach is not the same as exhibiting a short-term preference. Previous works [32, 44] suggests to adopt a procedure in which, in each cycle, the user selects the rule that is most similar to the target, i.e., $k = \arg\max_{i \in rec} sim\,(i, target)$. In our work, however, recommended items are intrinsically more complex than movies or songs, i.e., the "typical" target items of contemporary recommender systems. IF-THEN rules, indeed, are composite items that involve a trigger and an action, each one with its own details. Since users may not always be able to select the rule that is most similar to their target, we introduce a given amount of randomness (50%) in the user's choices. An investigation about the effects of introducing different amount of randomness is reported in Section 5.2.2). The simulated interaction stops when the target is one of the recommendations, i.e., $target \in rec$, or after 10 refinement cycles.

Thanks to the simulation procedure described above, we explore the effect of the following $HeyTAP^2$ characteristics over the computed metrics:

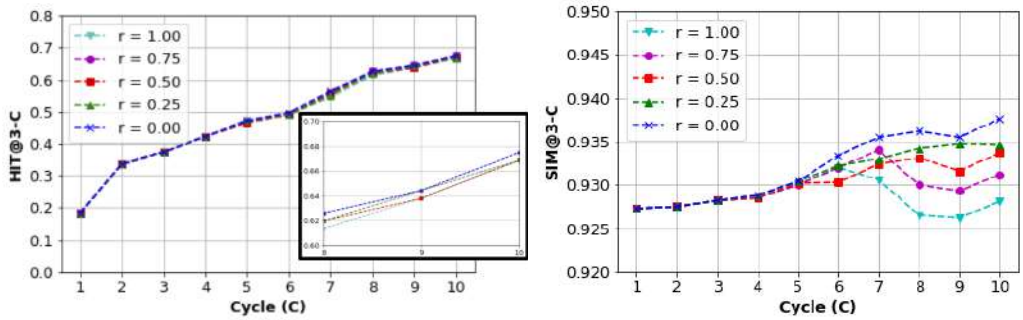
- The influence of the η parameter in Equation 5, i.e., the balance between the current personalization intention and the long-term preferences of the user. When $\eta = 0$, only the current intention is taken it account, while $\eta = 1$ means that $HeyTAP^2$ ignores the intention and computes recommendations by taking into account long-term preferences, only. As in[44], we vary η from 0 (intention only) to 1 (long-term preferences only) in steps of 0.25.
- The influence of the supported re-weighting policies Φ. These represent different ways of taking positive and negative feedback into account. At each refinement cycle, $HeyTAP^2$ can promote rules similar to the user's feedback and/or penalize the rules not selected by the user.

We also considered different values of θ in Equation 4 to extract long-term preferences. Here, we only show results for $\theta = 0.5$, i.e., the configuration that provided the best results in general.

5.2.2 Simulation Procedure Assessment. The adopted simulation procedure [32, 44] ideally assumes that, in each refinement cycle, the user is able to provide her short term preference by selecting the item that is more similar to her target. The complex nature of IF-THEN rules, however, may influence such an assumption. Figure 12a and Figure 12b show the impact of different amount of randomness in the user's short-term preferences (r, from 0% to 100%) on the **HIT@3** and **SIM@3** metrics, respectively. The figures refer to a generic configuration of $HeyTAP^2$ that takes into account both intentions and long-term preferences ($\eta = 0.5$), and that promotes *and* penalizes items ($\Phi = \Phi_{\pm}$).

As shown in Figure 12a, randomness has little or no influence on hit rate. After 10 refinement cycles, for instance, **HIT@3** is 0.675 without any random choices ($r = 0\%$), and 0.668 with random choices, only ($r = 100\%$). An increasing amount of randomness, instead, slightly decreases the similarity of the proposed recommendations with the target item as the interaction between the user and system proceeds (Figure 12b). After 10 refinement cycles, in particular, **SIM@3** ranges from 0.937 without any random choices ($r = 0\%$) to 0.928 with random choices, only ($r = 100\%$).

Overall, the limited effects of randomness suggest that the simulation procedure described in previous works [32, 44] can be effectively applied to our work. However, as a conservative choice, we specify the procedure to our domain by introducing an amount $r = 50\%$ of randomness in the



(a) HIT@3 (b) SIM@3

Fig. 12. The influence of different amount of randomness in the user's short-term preferences provided by the users in the refinement cycles.

refinement cycles. This allows the evaluation to account for situations in which the user fails to select the the item that is more similar to her target.

5.2.3 Results. Table 4 shows the results for the hit-rate and similarity metrics after 10 refinement cycles of $HeyTAP^2$ in different configurations. Columns are the different re-weighting policies Φ supported by $HeyTAP^2$, while rows refer to the different values of η. In general, $HeyTAP^2$ provided higher hit-rates and greater similarity with target items when either η is 0.50 or Φ is $\Phi_{\pm}$. The *best* performing configuration, underlined in the table, is at the intersection of those two conditions, i.e., when $\eta = 0.50$ *and* $\Phi = \Phi_{\pm}$. The re-weighting policy $\Phi_{\pm}$ means that candidate rules are promoted when are similar to the user's short-term preference and penalized when they are similar to the rules not chosen by the user in the refinement cycles. The value $\eta = 0.5$, instead, means $HeyTAP^2$ treats both intentions and long-term preferences at the same way.

	HIT@3-10			**SIM@3-10**		
η	Φ_+	Φ_-	$\Phi_{\pm}$	Φ_+	Φ_-	$\Phi_{\pm}$
0.00	0.626	0.632	0.638	0.925	0.925	0.927
0.25	0.663	0.656	0.663	0.931	0.934	0.934
0.50	0.669	0.669	**0.674**	0.933	0.935	**0.937**
0.75	0.497	0.503	0.503	0.776	0.780	0.783
1.00	0.233	0.233	0.233	0.561	0.561	0.561

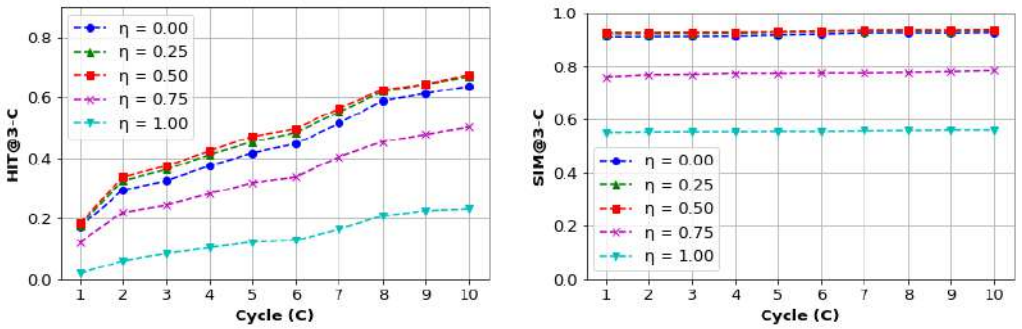
Table 4. The table shows the hit-rate and the similarity metric after 10 refinement cycles (HIT@3-10 and SIM@3-10, respectively). In general, $HeyTAP^2$ performed better with $\eta = 0.50$, i.e., by treating both intentions and long-term preferences at the same way, and $\Phi = \Phi_{\pm}$, i.e., by promoting and penalizing items. The best-performing configurations in terms of hit-rate and similarity, in particular, are underlined. Results are the average of 5 equals experiments.

For what concerns re-weighting policies, the table suggests that promoting (Φ_+) or penalizing (Φ_-) items, only, gives very similar results on the hit-rate and similarity metrics. Taking into account both positive and negative feedback at the same time ($\Phi_{\pm}$), instead, improves the recommendation process in terms of **HIT@3** and **SIM@3**. Furthermore, the table clearly highlights the importance

of the elicited personalization intention in the recommendations process. With $\eta \in [0, 0.5]$, i.e., from $HeyTAP^2$ considering only intentions ($\eta = 0$) to $HeyTAP^2$ mixing intentions with long-term preferences ($\eta = 0.5$), the hit-rate and similarity metrics are similarly promising: the hit-rate is greater than 0.6 and the similarity is greater than 0.9 independently of the adopted re-weighting policy Φ. As the importance of the intention decreases in favor of the users' long term preferences, i.e., $\eta \in (0.5, 1]$, both the hit-rate and the similarity with target items decreases consistently. However, results show that long-term preferences play an important role, too: mixing them with users' intentions ($\eta = 0.5$) allows $HeyTAP^2$ to increase its accuracy and its ability to recommend items similar to the users' target items. As reported in Table 5, in particular, taking into account long-term preferences allows users to get to their target item faster, i.e., with less refinement cycles on average.

η	M	SD
0.00	4.125	2.937
0.25	3.972	2.888
0.50	3.835	2.833
0.75	4.305	2.982
1.00	5.184	2.761

Table 5. The table shows how many refinement cycles on average are needed by $HeyTAP^2$ to recommended the user's target item, i.e., to generate a hit, for different values of η. Mixing long-term preference with personalization intentions ($\eta = 0.50$) allows users to get to their target items faster.



(a) HIT@3 in each recommendation cycle

(b) SIM@3 in each recommendation cycle

Fig. 13. Evolution of the hit-rate and similarity in 10 $HeyTAP^2$ refinement cycles with different values of η.

Figure 13 shows how the re-weighting policy $\Phi_{\pm}$ performs over 10 refinement cycles with different values for η in terms of hit-rate and similarity with the target item. To plot such metrics over time, we follow the procedure suggested in [44]. When the target item of a user is recommended in a recommendation set (rec) before the 10th cycle, i.e., there is a hit, we propagate the given rec to the subsequent cycles. This ensures that the computed metric values are always calculated over the same number of users. Figure 13a highlights that, independently of the η value, the hit-rate increases nearly linearly. Therefore, $HeyTAP^2$ allows the user to get closer to her target as the interaction proceeds. As the figure suggests, however, 10 refinement cycles are sometimes not

sufficient for a recommendation hit. A further investigation on the differences between hits and misses, i.e., target rules not hit after 10 refinement cycles, shows that the size of intention candidate rules sets (*IC*) is typically bigger with misses than hits, on average ($M = 205.78$, $SD = 186.24$ *vs.* $M = 169.04$, $SD = 184.54$, respectively). The size of the user's profile P, instead, is almost equal in misses and hits ($M = 66.39$, $SD = 75.22$ *vs.* $M = 65.19$, $SD = 63.96$, respectively). As an example, the following rule generated a miss: *"if I upload a video on YouTube, then create a video post on Tumblr."* Given the spread of social networks in different contexts, e.g., videos and photos, and their different posting mechanisms, the simulated intention for this rule can be mapped to 540 different IF-THEN rules in the exploited dataset. The rule *"if the WeMo motion sensor detects a movement, then set the fan mode on the WeMo humidifier"*, instead, generated a hit without any refinement cycle. In this case, the *WeMo* motion sensor and the *WeMo* humidifier are very specific connected entities with dedicated functionality, and, although abstract, the simulated intention can be associated to 32 IF-THEN rules involving other similar connected entities, only. This suggests that it is more difficult to get the recommendation process to a specific item when there are too many intention candidate rules: when this happens, even 10 refinement cycles may not be sufficient to prioritize the target item among the huge number of "similar" rules.

Figure 13 also shows that, differently from other *preference-based feedback* approaches [44] where the similarity of the recommendations with target items increased linearly, in $HeyTAP^2$ the **SIM@3** metric is high from the first refinement cycle, especially with $\eta \in [0, 0.5]$, and it remains roughly the same over the remaining cycles. By eliciting a personalization intention from the user, therefore, $HeyTAP^2$ is able to focus on IF-THEN rules that are specifically targeted for addressing the current user's need.

	PREC$_{ic}$@3-10			**REC$_{ic}$@3-10**		
η	Φ_+	Φ_-	$\Phi_\pm$	Φ_+	Φ_-	$\Phi_\pm$
0.00	0.975	0.975	0.975	0.142	0.142	0.142
0.25	0.975	0.975	0.975	0.142	0.142	0.142
0.50	0.975	0.975	0.973	0.142	0.142	0.142
0.75	0.689	0.688	0.688	0.101	0.100	0.100
1.00	0.344	0.344	0.344	0.050	0.050	0.050

Table 6. The table shows the precision and the recall metric over intention candidate rules after 10 refinement cycles (PREC$_{ic}$@3-10 and REC$_{ic}$@3-10, respectively). Independently of the adopted policy, $HeyTAP^2$ was able to suggest intention candidates rules, i.e., rules that satisfy the user's intention, even by mixing the intention with user's long-term preferences (see the table row with $\eta = 0.50$, for instance). Not surprisingly, such an ability decreases when long-term preferences are more important than intentions ($\eta = 0.75$ and $\eta = 1.00$). Results are the average of 5 equals experiments.

Finally, Table 6 further highlights how users' long term preferences influence the recommendation process. Independently of the adopted re-weighting policy, precision and recall over intention candidate rules are the same with $\eta \in [0, 0.5]$. This means that $HeyTAP^2$ is able to suggest intention candidate rules $\in IC$, i.e., rules that satisfy the current user's intention, even by mixing the intention with user's long-term preferences. As the table suggests, however, precision and recall decrease when long-term preferences become more important than the current intention in the recommendation process ($\eta = 0.75$ and $\eta = 1.00$).

5.2.4 Key Findings. Summarizing, the key findings resulting from our quantitative assessment of the recommendation process of $HeyTAP^2$ in different configurations (**RQ1**) are the following:

- hit rate increases as the interaction between $HeyTAP^2$ and the user proceeds;
- $HeyTAP^2$ performs better by taking into account both the current user's intention and her long-term preferences, as well as her positive and negative feedback;
- long-term preferences allow $HeyTAP^2$ to recommend the target item faster.

5.3 Comparison With Other Recommender Systems

To further assess the potential of our approach, we compare our CSR system with other recommender systems that are in the TAP domain or employ a similar *preference-based feedback* approach (**RQ2**). In presenting the results of such an evaluation, we highlight the peculiarities of $HeyTAP^2$, as well as its differences with the other evaluated recommender systems.

5.3.1 Experimental Setting. Given the specific context, i.e., trigger-action programming, and the characteristics of our approach, i.e., a recommendation process that works with semantic, unstructured item representations, we were only able to find two other state-of-the-art recommender systems to compare with $HeyTAP^2$, namely *RecRules* and *n-by-p*.

- *RecRules* [21] is, to our knowledge, the only other recommender system in the context of IF-THEN rules for personalizing connected entities. It employs a hybrid approach by integrating both content-based and collaborative information in a graph-based setting, and it extracts different types of features based on the underlying connections between graph's items to make top-N recommendations. As $HeyTAP^2$, it exploits semantic information, and, in particular, the EUPont ontology to descibe IF-THEN rules. Since *RecRules* is not conversational, we compare it with $HeyTAP^2$ without refinement cycles, only.
- *n-by-p* [44] is, to the best of our knowledge, the only other conversational recommender system for domains whose items have unstructured representations. It employs a *navigation-by-preference* approach, called "preference chain", similar to the $HeyTAP^2$ refinement cycles: given a seed item, it helps the user navigate through the item space by tacking into account the user's history and her short-term preferences. We implemented a version of the *n-by-p* system in Java by exploiting the best-performing configuration described in the original paper and by modeling IF-THEN rules with the same semantic information exploited in $HeyTAP^2$. To simulate the interaction between users and *n-by-p*, we follow the same procedure described by its authors. The procedure is similar to the one adopted for the $HeyTAP^2$ refinement cycles, except for the selection of the initial seed item, which is chosen randomly from the user's profile. Also for n-by-p, we introduce an amount $r = 50\%$ of randomness in the short-term preferences provided by users in the refinement cycles.

Since both *RecRules* and *n-by-p* cannot model personalization intentions, we choose, for our baseline, a configuration of $HeyTAP^2$ that takes into account users' long-term preferences, only, i.e., $\eta = 1$ and $\Phi = \Phi_{\pm}$, which were among the worst performing configurations for $HeyTAP^2$.

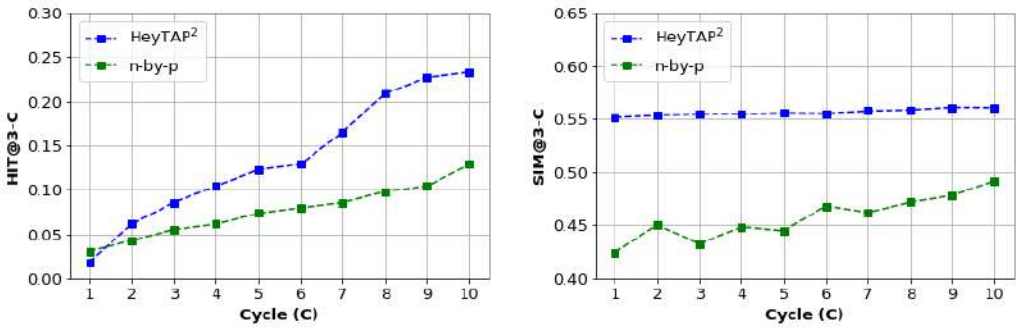
5.3.2 Results. Table 7 reports the results on the comparison of $HeyTAP^2$ with the two identified baselines, i.e., *RecRules* and *n-by-p*, in terms of hit-rate and similarity with the target item, while Table 8 focuses on precision and recall over intention candidate rules. We test all the three algorithms in the "single-shot" recommendation, i.e., without any refinement cycles ($c = 0$). Table 7 shows that, without refinement cycles and without taking into account personalization intentions, $HeyTAP^2$ is outperformed in the recommendation of the target item by the other two recommender system, with *RecRules* the best performing algorithm in terms of hit-rate. This is not surprising, and further confirms the importance of eliciting user's personalization intentions in our approach: without taking them into account, and without refinement cycles, a hybrid recommender system able to take into account both content-based and collaborative information like *RecRules* is clearly advantaged.

Given its content-based nature, however, *HeyTAP*^{2} provided recommendations similar to the target items from the beginning of the recommendation process (**SIM@3-0** = 0.552). The advantages of our approach can be seen, in Table 7, from the comparison of *HeyTAP*^{2} and *n-by-p* after 10 refinement cycles ("preference chain" in *n-by-p*). After some interactions between users and the systems, both the recommenders improve their hit-rate and similarity with target items. *HeyTAP*^{2}, however, provides better results.

RecSys	HIT@3-0	SIM@3-0	HIT@3-10	SIM@3-10
RecRules	**0.055**	0.057	✗	✗
n-by-p	0.031	0.425	0.110	0.426
HeyTAP^2	0.018	**0.552**	**0.233**	**0.561**

Table 7. Comparison of the *HeyTAP*^{2} recommendation process with *RecRules* [21] and with *n-by-p* [44] in terms of hit-rate and similarity with the target item. To compare *HeyTAP*^{2} with algorithms that do not take into account the user's intention, we run it in the configuration with long-term preferences, only, i.e., $\eta = 1$. We test all the three algorithm in the "single-shot" recommendation, i.e., without any refinement cycles ($c = 0$). Furthermore, we compare *HeyTAP*^{2} and *n-by-p* after 10 refinement cycles (what Rada et al. [44] call a "preference chain" in *n-by-p*). The best-performing algorithms are underlined. Results are the average of 5 equals experiments.

Such an advantage can be better glimpsed in Figure 14, in which we plotted the hit-rate and similarity metrics of *HeyTAP*^{2} and *n-by-p* over 10 recommendation cycles. Figure 14a, in particular, shows that while *n-by-p* starts with a higher hit-rate, the *HeyTAP*^{2} hit-rate increases faster and reaches a higher value, i.e., our recommendation process allows the user to get closer to her target more rapidly than *n-by-p*.



(a) HIT@3 in each recommendation cycle

(b) SIM@3 in each recommendation cycle

Fig. 14. Evolution of the hit-rate and similarity metrics in the comparison between *HeyTAP*^{2} and *n-by-p*.

Figure 14b, instead, highlights that the similarity of the recommendations with target items is constant in *HeyTAP*^{2}, while it increases nearly linearly in *n-by-p*, exactly as described in its original paper [44]. Furthermore, the *SIM*@3 metric is always higher in *HeyTAP*^{2}.

For what concerns precision and recall over intention candidate rules, Table 8 shows that *HeyTAP*^{2} is more inclined than the other two algorithms to recommend rules able to satisfy the user's intention, even without explicitly considering it. This suggests that users tend to define rules

RecSys	$\mathbf{PREC}_{ic}$@3-0	$\mathbf{REC}_{ic}$@3-0	$\mathbf{PREC}_{ic}$@3-10	$\mathbf{REC}_{ic}$@3-10
RecRules	0.086	0.001	✗	✗
n-by-p	0.184	0.002	0.150	0.022
$HeyTAP^2$	**0.337**	**0.005**	**0.344**	**0.050**

Table 8. Comparison of the $HeyTAP^2$ recommendation process with *RecRules* [21] and with *n-by-p* [44] in terms of precision and recall over intention candidate rules. As in Table 7, $HeyTAP^2$ considers long-term preferences, only, i.e., $\eta = 1$. The best-performing algorithms are underlined. Results are the average of 5 equals experiments.

that are semantically similar over time, thus further explaining why user's long-term preferences are an important factor to be considered in the recommendation process. Differently from *RecRules*, in particular, $HeyTAP^2$ is a pure content-based recommender system: while this may limit serendipity, i.e., the possibility of receiving "surprising" recommendations, it increases the ability of $HeyTAP^2$ to recommend rules that are semantically related to the user's need. Furthermore, the refinement cycles and the related short-term feedback adopted by $HeyTAP^2$ and *n-by-p* "drive" the recommendation process towards IF-THEN rules that can satisfy the current user's intention. As shown in Table 8, independently of the tested algorithm, both $\mathbf{PREC}_{ic}$ and $\mathbf{REC}_{ic}$ increase after 10 refinement cycles. As for the hit rate and the similarity metric, however, $HeyTAP^2$ outperformed *n-by-p*.

5.3.3 Key Findings. Summarizing, the key findings resulting from our comparison of $HeyTAP^2$ with state-of-the-art recommender systems (**RQ2**) are the following:

- two fundamental elements of the $HeyTAP^2$ recommendation process are the elicited user's intention and the refinement cycles: without them, hybrid recommeder systems like *RecRules* are more effective in terms of hit rate;
- $HeyTAP^2$ outperforms outperforms *n-by-p* after 10 refinement cycles in terms of hit rate and similarity of the recommendations with the target item;
- $HeyTAP^2$ is in general more inclined to recommend rules able to satisfy the user's intention with respect to the other evaluated algorithms, even without refinement cycles.

6 DISCUSSION

$HeyTAP^2$ is a semantic CSR system able to suggest pertinent and immediately applicable IF-THEN rules starting from an abstract description of a user's *personalization intention*. We evaluated the conversational and semantic recommendation process of $HeyTAP^2$ through different experiments with simulated users and real-world data contained in a dataset of IF-THEN rules extracted from IFTTT [53], one of the most popular TAP platforms. We decided to focus on the recommendation process given its importance in the overall system. Indeed, the other part of the system, i.e., the conversational agent to elicit users' intentions, exploits established technologies such as DialogFlow, and it is based on the results of a formative study conducted in our previous work [22].

The assessment of the recommendation process in its several configurations highlighted the advantages of the different characteristics of our approach, and allowed us to define the winning configuration. As expected, one of the most influencing factor in $HeyTAP^2$ was the user's personalization intention. The introduction of such an information in the recommendation process increases the recommendation accuracy and the similarity of the computed recommendations with the users' target items. However, also users' long-term preferences play an important role, and, if *mixed* with the users' intentions, allow $HeyTAP^2$ to achieve better results faster, i.e., with less refinement cycles. While, intuitively, users would like to find "new" rules rather than ones similar

to what they have already interacted with, the importance of the long-term preferences can be qualitatively explained by the fact that the "new" concept in trigger-action programming is different than in other domains where recommender systems are popular, e.g., movies. Rather than receiving "serendipitous" recommendations involving yet unknown triggers, actions, and connected entities, indeed, users typically have abstract personalization goals that can be addressed through multiple IF-THEN rules at run-time [20, 52]. As demonstrated by our investigation of precision and recall over intention candidate rules, this means users tend to define rules that are "semantically" similar over time, e.g., to adapt their rules to different contextual situations, and further explains why it is worth considering long-term preferences in the recommendation process. We also demonstrated the advantage of taking into account both positive and negative feedback in the refinement cycles. Promoting items similar to the users' short-term preferences while penalizing items similar to the "discarded" rules ($\Phi_{\pm}$), indeed, turned out to be the best-performing re-weighting policy Φ.

In comparing $HeyTAP^2$ with baseline recommender systems, we found interesting results. First, we further confirmed the peculiarity of our approach, i.e., eliciting users' intentions and using them to "guide" the recommendation process. To compare our system with algorithms that do not model such a concept, indeed, we used an $HeyTAP^2$ version that considered long-term preferences, only. Without intentions and refinement cycles, the *RecRules* algorithm, i.e., a recommender system that is explicitly designed for not "guided" and "single-shot" recommendations, outperformed $HeyTAP^2$. This can be explained by the fact that, besides long-term preferences extracted from the user's profile, *RecRules* can also exploit collaborative information extracted from other users. While approaches like *RecRules* can promote serendipity by making users discover new and interesting IF-THEN rules to be activated, however, they are less useful for assisting users that already have a goal, i.e., an intention, to fulfill. With refinement cycles, moreover, even the $HeyTAP^2$ version without intentions achieved promising results, and, in particular, performed better than the other evaluated *preference-based feedback* approach, i.e., *n-by-p*. As the interaction between the user and the systems proceeded, for instance, the hit-rate with the target item increased in both the recommenders. The $HeyTAP^2$ hit rate, however, increased faster, by reaching a higher value. Furthermore, the similarity of the recommendations with target items was always higher in $HeyTAP^2$. A possible explanation lies in the differences of the two approaches in calculating the scores to rank recommendations. In *n-by-p*, in fact, item's features are used *indirectly*, only, i.e., by computing scores on the basis of the "neighbors" of a given rule. In $HeyTAP^2$, instead, features are used directly, and scores reflect the Jaccard similarity of the rule's features.

6.1 Limitations

We are aware that the results presented in this paper could depend on numerous factors, including the high degree of sparsity of the evaluated dataset. Unfortunately, differently from other domains such as movies and songs, recommendations in the trigger-action programming context are in their early stages. To our knowledge, the dataset of Ur et al. [53] is the only publicly available collection of IF-THEN rules. Furthermore, other popular platforms such as Zapier and Microsoft Flow do not allow users to share their rules, making it impossible to crawl data from them. Even if further investigation is needed, however, results of the $HeyTAP^2$ evaluation are promising, and call for a new bread of conversational TAP platforms that guide their users in translating their abstract needs into executable IF-THEN rules.

7 CONCLUSIONS AND FURTHER DEVELOPMENTS

In the Internet of Things era, users can personalize the behavior of their connected entities by defining IF-THEN rules such as "if the entrance *Nest* security camera detects a movement, then blink the *Philips Hue* lamp in the kitchen." Unfortunately, the spread of new supported technologies

make the number of possible combinations between triggers and actions continuously growing, and trigger-action programming becomes a complex task. In this paper, we presented $HeyTAP^2$, a semantic CSR system to simplify the process needed by end users to translate their abstract needs into executable IF-THEN rules. The user can communicate her current personalization intention to a conversational agent to get a first set of recommended IF-THEN rules. Then, she can collaborate with the $HeyTAP^2$ through multiple refinement cycles to get recommendations that better align with her intention. By evaluating $HeyTAP^2$ with simulated users and real-word data extracted from IFTTT, we explored the advantages of the different configuration of our approach, and we successfully compared it with baseline recommender systems.

While further studies with real users and connected entities are needed to confirm our results, this paper open the way to novel TAP platforms able to *converse* with their users, *elicit* their abstract personalization intentions, and *recommend* pertinent and immediately applicable IF-THEN rules.

REFERENCES

[1] 2019. IFTTT. https://ifttt.com/ Accessed: 2019-11-20.

[2] 2019. Microsoft Flow. https://flow.microsoft.com/en-us/ Accessed: 2019-11-20.

[3] 2019. Zapier. https://zapier.com/ Accessed: 2019-11-20.

[4] 2020. Amazon Alexa. https://developer.amazon.com/en-US/alexa Accessed: 2020-01-20.

[5] 2020. Google Assistant. https://developers.google.com/assistant Accessed: 2020-01-20.

[6] Xavier Amatriain, Josep M. Pujol, Nava Tintarev, and Nuria Oliver. 2009. Rate It Again: Increasing Recommendation Accuracy by User Re-Rating. In *Proceedings of the Third ACM Conference on Recommender Systems* (New York, New York, USA) *(RecSys '09)*. Association for Computing Machinery, New York, NY, USA, 173–180. https://doi.org/10.1145/1639714.1639744

[7] Sarabjot Singh Anand, Patricia Kearney, and Mary Shapcott. 2007. Generating Semantically Enriched User Profiles for Web Personalization. *ACM Transactions on Internet Technology* 7, 4, Article 22 (Oct. 2007). https://doi.org/10.1145/1278366.1278371

[8] Grigoris Antoniou, Paul Groth, Frank van van Harmelen, and Rinke Hoekstra. 2012. *A Semantic Web Primer*. The MIT Press.

[9] We are Social. 2020. Digital in 2020. https://wearesocial.com/blog/2020/01/digital-2020-3-8-billion-people-use-social-media.

[10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A Survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking* 54 (15 October 2010), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010

[11] Barbara Rita Barricelli and Stefano Valtolina. 2015. *End-User Development: 5th International Symposium, IS-EUD 2015, Madrid, Spain, May 26-29, 2015. Proceedings.* Springer International Publishing, Cham, Germany, Chapter Designing for End-User Development in the Internet of Things, 9–24. https://doi.org/10.1007/978-3-319-18425-8_2

[12] C. Bizer and T. Heathand T. Berners-Lee. 2009. Linked data - the story so far. *International Journal on Semantic Web and Information System* 5, 3 (2009), 1–22.

[13] Julia Brich, Marcel Walch, Michael Rietzler, Michael Weber, and Florian Schaub. 2017. Exploring End User Programming Needs in Home Automation. *ACM Transaction on Computer-Human Interaction* 24, 2, Article 11 (April 2017), 35 pages.

[14] Derek Bridge, Mehmet H. Göker, Lorraine McGinty, and Barry Smyth. 2005. Case-Based Recommender Systems. *Knowledge Engineering Review* 20, 3 (Sept. 2005), 315–320. https://doi.org/10.1017/S0269888906000567

[15] Robin Burke. 2000. Knowledge-Based Recommender Systems. In *Encyclopedia of Library and Information Systems*. Marcel Dekker, 2000.

[16] Iván Cantador, Alejandro Bellogín, and Pablo Castells. 2008. A Multilayer Ontology-based Hybrid Recommendation Model. *AI Communications* 21, 2-3 (April 2008), 203–210.

[17] V. Cerf and M. Senges. 2016. Taking the Internet to the Next Physical Level. *IEEE Computer* 49, 2 (Feb 2016), 80–86. https://doi.org/10.1109/MC.2016.51

[18] Li Chen and Pearl Pu. 2012. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction* 22, 1 (2012), 125–150. https://doi.org/10.1007/s11257-011-9108-6

[19] F. Corno, L. De Russis, and A. Monge Roffarello. 2017. A Semantic Web Approach to Simplifying Trigger-Action Programming in the IoT. *Computer* 50, 11 (2017), 18–24. https://doi.org/10.1109/MC.2017.4041355

[20] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. A high-level semantic approach to End-User Development in the Internet of Things. *International Journal of Human-Computer Studies* 125 (2019), 41 – 54. https:

//doi.org/10.1016/j.ijhcs.2018.12.008

[21] 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* 10, 5, Article 58 (Sept. 2019), 27 pages. https://doi.org/10.1145/3344211

[22] 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 2020 International Conference on Advanced Visual Interfaces* (Ischia, Italy) *(AVI '20)*. Association for Computing Machinery, New York, NY, USA. To appear.

[23] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2020. TAPrec: Supporting the Composition of Trigger-Action Rules through Dynamic Recommendations. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (Cagliari, Italy) *(IUI '20)*. Association for Computing Machinery, New York, NY, USA, 579–588. https://doi.org/10.1145/3377325.3377499

[24] Florian Daniel and Maristella Matera. 2014. *Mashups: Concepts, Models and Architectures*. Springer Publishing Company, Incorporated.

[25] Luigi De Russis and Alberto Monge Roffarello. 2020. Personalizing IoT Ecosystem via Voice. In *EMPATHY: Empowering People in Dealing with Internet of Things Ecosystems* (Ischia, Italy) *(AVI '20)*. To appear.

[26] 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* 24, 2, Article 12 (2017), 52 pages. https://doi.org/10.1145/3057859

[27] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive Prototyping of Context-aware Applications. In *Proceedings of the 4th International Conference on Pervasive Computing* (Dublin, Ireland) *(PERVASIVE'06)*. Springer-Verlag, Berlin, Heidelberg, 254–271. https://doi.org/10.1007/11748625_16

[28] Gerhard Fischer. 2009. End-User Development and Meta-design: Foundations for Cultures of Participation. In *End-User Development*, Volkmar Pipek, Mary Beth Rosson, Boris de Ruyter, and Volker Wulf (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 3–14.

[29] Mathias Funk, L.-L Chen, S.-W Yang, and Y.-K Chen. 2018. Addressing the need to capture scenarios, intentions and preferences: Interactive intentional programming in the smart home. *International Journal of Design* 12 (04 2018), 53–66.

[30] Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural Approaches to Conversational AI. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1371–1374. https://doi.org/10.1145/3209978.3210183

[31] 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, 2, Article 14 (2017), 33 pages. https://doi.org/10.1145/3057861

[32] Lorraine Mc Ginty and Barry Smyth. 2002. Comparison-Based Recommendation. In *Advances in Case-Based Reasoning*, Susan Craw and Alun Preece (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 575–589.

[33] Will Haines, Melinda Gervasio, Aaron Spaulding, and Bart Peintner. 2010. Recommendations for End-User Development. In *Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI)*.

[34] 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* (Osaka, Japan) *(UbiComp '15)*. Association for Computing Machinery, New York, NY, USA, 215–225. https://doi.org/10.1145/2750858.2805830

[35] Ting-Hao K. Huang, A. Azaria, and J. P. Bigham. 2016. InstructableCrowd: Creating IF-THEN Rules via Conversations with the Crowd. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (Santa Clara, California, USA) *(CHI EA '16)*. ACM, New York, NY, USA, 1555–1562. https://doi.org/10.1145/2851581.2892502

[36] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2020. A Survey on Conversational Recommender Systems. arXiv:2004.00646 [cs.HC]

[37] Joseph A. Konstan and John Riedl. 2012. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction* 22, 1 (2012), 101–123. https://doi.org/10.1007/s11257-011-9112-x

[38] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. *End User Development*. Springer Netherlands, Dordrecht, Netherlands, Chapter End-User Development: An Emerging Paradigm, 1–8. https://doi.org/10.1007/1-4020-5386-X_1

[39] Friedemann Mattern and Christian Floerkemeier. 2010. *From the Internet of Computers to the Internet of Things*. Springer Berlin Heidelberg, Berlin, Heidelberg, 242–259. https://doi.org/10.1007/978-3-642-17226-7_15

[40] 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* (London, United Kingdom) *(IMC*

'17). ACM, New York, NY, USA, 398–404. https://doi.org/10.1145/3131365.3131369

[41] Dejan Munjin. 2013. *User Empowerment in the Internet of Things*. Ph.D. Dissertation. Université de Genève. http://archive-ouverte.unige.ch/unige:28951

[42] Abdallah Namoun, Athanasia Daskalopoulou, Nikolay Mehandjiev, and Zhang Xun. 2016. Exploring Mobile End User Development: Existing Use and Design Factors. *IEEE Transactions on Software Engineering* 42, 10 (Oct 2016), 960–976. https://doi.org/10.1109/TSE.2016.2532873

[43] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. 2016. SPrank: Semantic Path-Based Ranking for Top-N Recommendations Using Linked Open Data. *ACM Trans. Intell. Syst. Technol.* 8, 1, Article 9 (Sept. 2016), 34 pages. https://doi.org/10.1145/2899005

[44] Arpit Rana and Derek Bridge. 2020. Navigation-by-Preference: A New Conversational Recommender with Preference-Based Feedback. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (Cagliari, Italy) *(IUI '20)*. Association for Computing Machinery, New York, NY, USA, 155–165. https://doi.org/10.1145/3377325.3377496

[45] Ian Ruthven and Mounia Lalmas. 2003. A Survey on the Use of Relevance Feedback for Information Access Systems. *Knowl. Eng. Rev.* 18, 2 (June 2003), 95–145. https://doi.org/10.1017/S0269888903000638

[46] Giovanni Semeraro, Pasquale Lops, Pierpaolo Basile, and Marco de Gemmis. 2009. Knowledge Infusion into Content-based Recommender Systems. In *Proceedings of the Third ACM Conference on Recommender Systems* (New York, New York, USA) *(RecSys '09)*. ACM, New York, NY, USA, 301–304. https://doi.org/10.1145/1639714.1639773

[47] Barry Smyth. 2007. *Case-Based Recommendation*. Springer-Verlag, Berlin, Heidelberg, 342–376.

[48] Barry Smyth and L McGinty. 2003. An Analysis of Feedback Strategies in Conversational Recommenders.. In *Proceedings of the Fourteenth Irish Artificial Intelligence and Cognitive Science Conference*.

[49] Barry Smyth and Lorraine McGinty. 2003. The Power of Suggestion. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (Acapulco, Mexico) *(IJCAI'03)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 127–132.

[50] Kathryn T. Stolee and Sebastian Elbaum. 2013. Identification, Impact, and Refactoring of Smells in Pipe-Like Web Mashups. *IEEE Transactions on Software Engineering* 39, 12 (Dec 2013), 1654–1679. https://doi.org/10.1109/TSE.2013.42

[51] Quan Thanh Tho, Siu Cheung Hui, A. C. M. Fong, and Tru Hoang Cao. 2006. Automatic Fuzzy Ontology Generation for Semantic Web. *IEEE Transaction on Knowledge and Data Engineering* 18, 6 (June 2006), 842–856. https://doi.org/10.1109/TKDE.2006.87

[52] 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* (Toronto, Ontario, Canada) *(CHI '14)*. ACM, New York, NY, USA, 803–812. https://doi.org/10.1145/2556288.2557420

[53] 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* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3227–3231. https://doi.org/10.1145/2858036.2858556

[54] Pierre-Yves Vandenbussche, Ghislain Atemezing, María Poveda-Villalón, and B. Vatant. 2017. Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web. *Semantic Web* 8 (01 2017), 437–452. https://doi.org/10.3233/SW-160213

[55] Lina Yao, Quan Z. Sheng, Anne H.H. Ngu, Helen Ashman, and Xue Li. 2014. Exploring Recommendations in Internet of Things. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (Gold Coast, Queensland, Australia) *(SIGIR '14)*. ACM, New York, NY, USA, 855–858. https://doi.org/10.1145/2600428.2609458

[56] Lefan Zhang, Weijia He, Jesse Martinez, Noah Brackenbury, Shan Lu, and Blase Ur. 2019. AutoTap: Synthesizing and Repairing Trigger-Action Programs Using LTL Properties. In *Proceedings of the 41st International Conference on Software Engineering* (Montreal, Quebec, Canada) *(ICSE '19)*. IEEE Press, 281–291. https://doi.org/10.1109/ICSE.2019.00043

[57] 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, Article 104 (Sept. 2020), 26 pages. https://doi.org/10.1145/3411838

[58] Shiyu Zhang, Juan Zhai, Lei Bu, Mingsong Chen, Linzhang Wang, and Xuandong Li. 2020. Automated Generation of LTL Specifications for Smart Home IoT Using Natural Language. In *Proceedings of the 23rd Conference on Design, Automation and Test in Europe* (Grenoble, France) *(DATE '20)*. EDA Consortium, San Jose, CA, USA, 622–625.

[59] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. 2018. Towards Conversational Search and Recommendation: System Ask, User Respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 177–186. https://doi.org/10.1145/3269206.3271776