Evaluating Rule-based Programming and Reinforcement Learning for Personalising an Intelligent System

Ruixue Liu
WPI
Worcester, MA, USA
rliu2@wpi.edu

Advait Sarkar
Microsoft Research
Cambridge, UK
advait@microsoft.com

Erin Solovey
WPI
Worcester, MA, USA
esolovey@wpi.edu

Sebastian Tschiatschek
Microsoft Research
Cambridge, UK
setschia@microsoft.com

ABSTRACT

Many intelligent systems can be personalised by end-users to suit their specific needs. However, the interface for personalisation often trades off the degree of personalisation achievable with time, effort, and level of expertise required by the user. We explore two approaches to end-user personalisation: one asks the user to manually specify the system's desired behaviour using an end-user programming language, while the other only asks the user to provide feedback on the system's behaviour to train the system using reinforcement learning. To understand the advantages and disadvantages of each approach, we conducted a comparative user study. We report participant attitudes towards each and discuss the implications of choosing one over the other.

CCS CONCEPTS

• **Human-centered computing → User interface programming**.

KEYWORDS

reinforcement learning, end-user programming.

ACM Reference Format:
Ruixue Liu, Advait Sarkar, Erin Solovey, and Sebastian Tschiatschek. 2019.Evaluating Rule-based Programming and Reinforcement Learning for Personalising an Intelligent System. In *Joint Proceedings of the ACM IUI 2019 Workshops, Los Angeles, USA, March 20, 2019*, 11 pages.

1 INTRODUCTION

Motivation: Eyes-free Participation in Meetings

Entering a meeting room, attendees immediately gain contextual information about the meeting, including who is already in the room and what they are doing. It has been shown that this information is crucial to meeting participation [5, 8, 13, 16], and that without it, a meeting attendee can be more inhibited, and less able to contribute to the meeting. For a number of reasons, participants of modern meetings may not have access to this information. For instance, blind and low vision individuals have described this information asymmetry as a major hurdle in meetings [13, 22]. Participants often join meetings remotely without access to video, due to bandwidth limitations, or due to a parallel eyes-busy task such as driving or cooking [13, 14, 16].

To facilitate *eyes-free* participation in a meeting, we are studying the use of a computer vision system to extract the visual information that is important for equitable meeting participation. Information that may be extracted include attendee identity [21], pose [23], location, focus of attention [17], and other visual features such as clothing [9]. Computer vision systems might even be able to infer estimated age and gender [17, 19]. Any information thus inferred could be relayed to the user via eyes-free methods such as sonifications, text-to-speech, or tactile output. Given that a new entrant to a meeting only has a small amount of time available to gain context, it is important that the most useful subset of available information is presented to the user.

The 'most useful subset' varies not only from context to context, but also from user to user. A goal of an intelligent system is to automatically adapt to the context or to users' preferences to achieve a personalised experience and improve performance. It is not possible for system designers to correctly anticipate user preferences in many contexts, so a naïve heuristic-based solution to the subset selection problem would not be satisfactory. Thus, it is important to have a personalization interface. However, it is challenging to enable users to specify preferences for automated adaptation without requiring excessive time and effort.

Personalisation Approaches

This paper explores two alternatives for end-user personalisation of such a system. One approach is to learn automatically from how the user interacts with the system and from user feedback, which is well suited to reinforcement

IUI Workshops'19, March 20, 2019, Los Angeles, USA

learning models. Reinforcement learning typically requires a large number of feedback instances from the user to reach the desired level of personalisation. The reward signal for reinforcement can be given explicitly (e.g., by the user marking whether behaviour was appreciated or not) or implicitly (e.g., if the user turns the volume down for some notifications, or actively queries the system for information). End-users may not be able to give a fully-informed response because they cannot visually perceive all information in the environment, and may not realise when insufficient or incorrect information has been given. This further leads to users having less control and therefore less ownership and trust of the system.

An alternative approach is to enable the user to directly specify their preferences, using a simple end-user programming paradigm to define the system's behaviour, such as rule-based programming (e.g., if this, then that). Explicit rule-based programming has the potential for avoiding user frustration and increasing ownership of the system. However, users might struggle to specify their needs in such ways; or to anticipate what information they may want in different situations. To design such systems effectively, we were motivated to understand strengths and weaknesses of each approach. Background work in this area is in Appendix B.

2 MEETING SIMULATOR IMPLEMENTATION

To explore personalisation for an eyes-free meeting, we created a meeting simulator. It creates a readily-available source of well-controlled meeting scenarios. Each generated meeting contains a variable number of *attendees*. For each attendee, the following 8 attributes are generated: *relation*, *name*, *pose*, *activity*, *eye gaze mode*, *location*, *clothing* and *gender*. These are further elaborated in Table 1 in Appendix A. These were chosen based on current inference capabilities of computer vision systems, as well to provide enough diversity for interesting personalisation opportunities.

Each simulated meeting contained 3–8 attendees. Participants provided names of colleagues prior to the study, allowing us to constrain each simulated meeting to contain 1–3 attendees known to the participant. The value for each attribute was randomly chosen, with constraints. For example, if an attendees' *activity* is typing, then the *eye gaze mode* will be looking at the phone or laptop.

Two interfaces were built to personalise meeting overviews, corresponding to rule-based programming and reinforcement learning. Both used the same meeting attributes as the basis for personalised notifications.

Interface 1: Rule-based programming

This interface supports the manual creation of different sets of rules for different global contexts (e.g. different meeting sizes). The user would define the global context first (e.g. total number of attendees > 5). Then they would define the associated notifications (e.g. provide the *name* for all attendees). They could have a different set of rules for a different context (e.g. if the total number of attendees less than or equal to five, provide the *name* and *location* for all attendees). Alternately, a user may want different notifications based who is speaking. Then one possible set of rules could be: when there is an attendees whose *relation* is known and *activity* is speaking, tell me the *eye gaze mode* for all attendees; when there is an attendee whose relation is unknown and activity is speaking, tell me the *pose* for all attendees. The second consideration is to set notifications for specific attendees. For example, the user might want to know different information for attendees based on their *relation*. Possible rules could be: tell me the *name* and *activity* for attendees whose *relation* is known, tell me the *location* and *pose* for attendees whose *relation* is unknown.

Based on this, we divide our rule-based programming interface into two parts to support two level nesting if-clauses: the if-clause to set the 'global context' condition, and the if-clause to set notifications for specific attendees when the first-cause is satisfied (Figure 1). The first part enables users to set the 'global context' condition for the whole meeting, which could be: when the total number of all attendees $\{=, <, > x\}$, x is a number between 0 and 8; when the total number of attendees whose *one attribute is or is not one certain value* $\{=, <, > x\}$; when there is an attendee whose *one attribute is or is not one certain value* and *another attribute is or is not one certain value*. The second part is setting notifications for specific attendees when the meeting satisfies the 'context' condition, which could be: if attendees' *one attribute is or is not one certain value*, then tell me *these attributes*.

Users can enter an arbitrary number of rules, and an arbitrary number of if-clauses and notifications per rule. Users can also review, edit and delete existing rules (Figure 1).

Interface 2: Reinforcement learning

In reinforcement learning [20] an agent interacts with an environment by taking actions with the goal of accumulating reward. The interaction is in discrete steps: in step t, the environment is in state s_t and the agent takes action a_t; it then receives information about the next state s_{t+1} of the environment and the obtained reward r_t which typically depends on both s_t and a_t. Initially, the agent does not know the environment's dynamics (state transitions and reward) but can learn about these from interactions with the environment. It can then adapt its behaviour to maximize cumulative reward.

Reinforcement learning problems can be solved using Q-learning and an ϵ-greedy policy [20]. We implemented the deep learning approach from [15] for Q-learning. This approach uses a neural network (Q-network) that takes a featurised state representation as input and maps it to the corresponding Q-values—from those we can identify the best

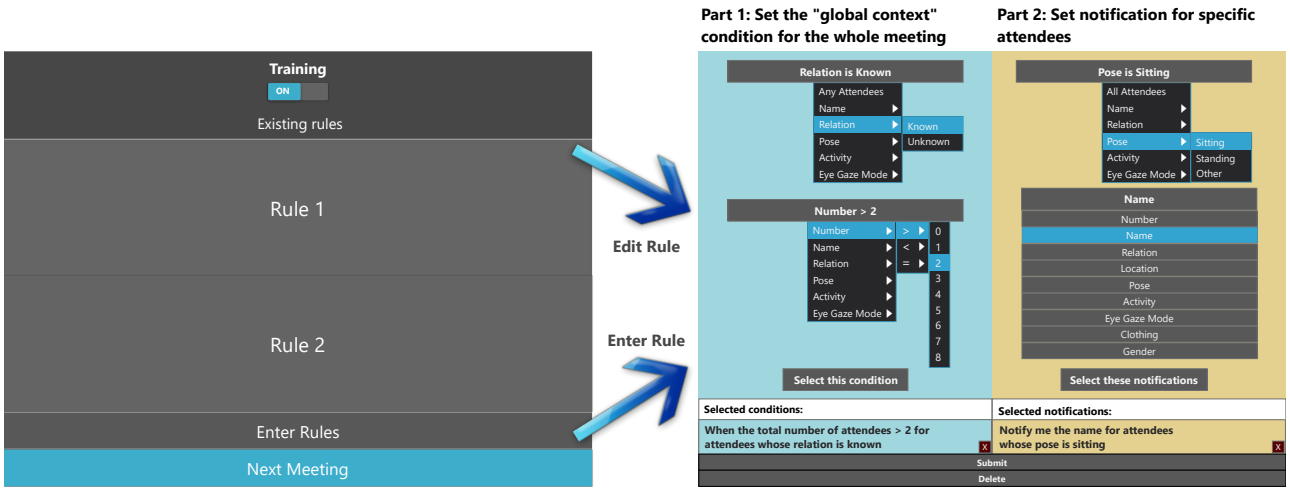


Figure 1: Interface for rule-based programming. Users can create different meeting contexts (Part 1). For each context, a set of notification rules can be defined (Part 2). For example, there may be different notification rules based on meeting size (context).

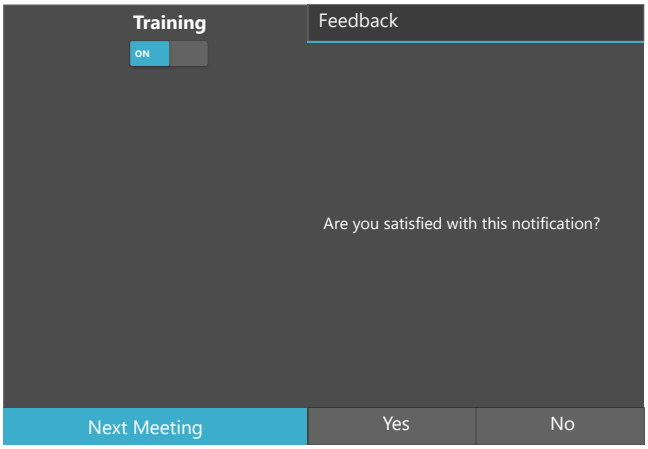


Figure 2: Participants used this interface to provide feedback for reinforcement learning after each notification.

action to take.Typically, training such models requires huge amounts of feedback. With the goal of being parsimonious in terms of required feedback, we designed the reinforcement learning model to select notifications for each attendee individually. We only allowed users to give binary feedback (a reward of ±1) corresponding to whether a user was satisfied with the notifications about each attendee (Figure 2).

In the context of our meeting simulator, the input to the Q-network includes the features of each attendee, which are the values of each attribute. Each instance of the attendee's attributes used in the rule-based programming interface is a feature for the reinforcement learning model. There are 17 features in total, which includes 4 features for *name* (known A, known B, known C, other), 2 features for *relation* (known, unknown), 3 features for *pose* (sitting, standing, other), 5 features for *activity* (reading, typing, speaking, listening, other), and 3 features for *eye gaze mode* (screen projector, other member in meeting, other). The Q-network consists of three fully connected (FC) layers with rectified linear unit (ReLu) activation functions [6] and 200 neurons each. We used an ϵ-greedy policy of ϵ=0.2, and we implemented an optimizer using RMSprop with a learning rate of 0.001. The possible actions correspond to which attributes of the meeting attendee the user should be notified about. The 8 attributes result in 256 possible actions (2^8=256).

3 USER STUDY

We focus here specifically on the moments at the very *beginning* of a meeting, which we call the *meeting overview*, rather than an ongoing assistive experience throughout the meeting. While we recognise the importance of ongoing assistance, we have chosen to focus only on the meeting overview to make the scope of our study tractable.

Participants

Fifteen participants were recruited using convenience sampling, 8 female and 7 male. Participants spanned four organizations and worked in a range of professions including real estate planning and surveying, business operations, interaction design research, machine learning research, and biomedical research. Six participants self defined as programmers, and among them, five participants had experience in machine learning from taking courses or research experience. We will refer to our participants as P1, P2, ..., P15.

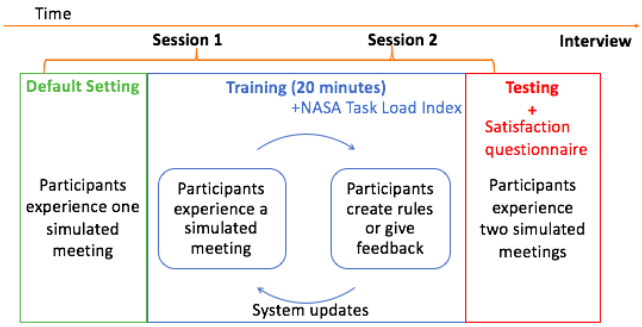


Figure 3: Experimental procedure

Scenario and task

In this experiment, we considered a scenario where participants need to join a meeting with a few colleagues that they know and some people that they do not know. We told the participants that they had dialed into the meeting late without any video, but that the system could give them an overview of what was happening when they entered the meeting. They were then given a detailed explanation of the various attributes. The task was to train the system to only deliver information of interest to the participant.

Procedure

The experiment used a within-subject design. Our two conditions were *rule-based programming* and *reinforcement learning*. All participants performed the task twice, once in each condition, in counterbalanced order.

Participants were first introduced to the meeting simulator. Then there were two sessions. In each session, participants used a different adaption technique. There were 3 sections in one session: default setting, training, and testing. Figure 3 describes this procedure and timeline. In the 'default setting' section, participants entered a simulated meeting, where they were notified about *all information* about all attendees.

In each 'training' section, participants were asked to use the two strategies we provide to personalise the notification system for 20 minutes to train the system to only give them the information they want to know to understand the situation. The participants' cognitive load for this training process was measured using the standardised NASA Task Load Index questionnaire ([7]). Then, in the 'testing' section, participants entered another two simulated meetings, where their own personalised notification system was used.

Participants also filled out a questionnaire about their satisfaction level with using this approach. This questionnaire consists of four questions on a 7-point scale. Q1: after test meeting 1, how relevant was the notification to your interest? Q2: after test meeting 2, how relevant was the notification to your interest? Q3: How successful were you at creating the rules/how successful do you think the system was at learning your preferences? Q4: In general, how satisfied are you with the notifications after training the system?

At the end of the experiment, we conducted a short semi-structured interview to ask about participants' experiences and opinions. We asked what participants struggled with when using the two techniques, their opinions about the advantages and disadvantages of the two techniques they experienced, as well as what would be the ideal approach for them to personalise such a system. Participants were encouraged to think-aloud throughout the experiment. The entire study was audio recorded and transcribed.

For the rule-based programming approach, the researcher also helped participants navigate the interface. The researcher did not help participants with developing rules or formulating rules. Considering the time constraint of the experiment, this enabled participants to focus on the rule-based approach in comparison with the reinforcement learning approach, rather than the specific interface that may require training to fully understand. The experiment took about 60 minutes and participants were compensated £20 for their time.

4 RESULTS

Our quantitative comparisons showed that participants reported a significantly lower cognitive load, and significantly higher satisfaction scores, when using rule-based programming rather than reinforcement learning. Our complete statistical analysis is presented in Appendix C.

The rest of this section presents our qualitative analysis of participants' experience during and after the experiment.

Rule-based programming

Even though all participants were able to personalise the system by using rule-based programming, they also asked questions or requested help from the researcher. The following sections present common issues participants experienced.

Difficulty understanding the filtered information. Some participants had difficulty understanding the meetings with the notifications constrained by their rules. This means that participants had false expectations about the notifications they would receive based on their rules, and therefore had difficulties uncovering what was happening in the meeting based on the filtered information. This issue always happened at the beginning of the training, and was always resolved with the help of the researcher. For example, P5 described her frustrations: *"I want to know when people are speaking, whether people are listening and what they are doing now, this includes much more information. For example, when I imagine this, I thought I would know the people who are speaking and who are listening, and what exactly they are doing. When I told you to use the rules, it lost too much information."*

In this case, P5 entered rules to set the notifications to tell her who is speaking and who is listening, but since there was no one speaking or listening in the meeting overview (the beginning of the meeting), there was no notification. However, P5 had difficulty understanding this logic.

Difficulty maintaining multiple rules. While the system supports participants in creating multiple rules, it is also more difficult to keep track of multiple rules and understand how they work together. For example, P2 asked the researcher when creating multiple rules: *"So all of these rules are stacking up?"* P4 questioned how multiple rules work together after trying multiple rules in one meeting: *"Okay, the last one was very short, did some of these [rules] override each other?"*

Participants also became confused with notifications when dealing with multiple rules, even when knowing how they work together. For example, P6 created different rules for meetings with different sizes. After one meeting, P6 mentioned he was not satisfied with the information. The researcher inquired further, and mentioned that the meeting is a small size and the notifications were given based on his rules for small meetings. Then, P6 said: *"This is a small meeting? Oh, then yes, I am satisfied. Sometimes I get confused."*

Confusion about what the interface supports. Some participants expressed confusion about the capability of the interface. For example, P2 said, *"I don't know [if] what I need, or what I am asking for, is more granularity in control or things which can't be given."* Some participants wanted functionality the interface did not provide. For example, P3 commented, *"The way I was hoping is to be able to train the system to take out the stuff I don't like to hear about, like I don't want to know the clothing for everyone."* However, the interface only supports users to specify attributes they want to hear. Some participants were unaware of functions the interface supported. P11 commented, *"Still there some things, instances or features, like other activities, are not that informative, but I guess it is the best we can do, I can not just grows out the other activity."* Although, in fact, the interface can support users to specify *if the activity is other, then...*

Reinforcement learning

Difficulty giving binary feedback. Arriving at a binary judgment about each notification was described as difficult by many participants. Frequently, they expressed a desire to be able to give richer feedback. In keeping with best practice for fast convergence of reinforcement learning, participants were instructed to only give positive feedback to the system when they were completely satisfied with the notification, i.e., when there is neither any unnecessary nor missing information. However, participants struggled to understand this concept and usually needed further explanation. For example, P10 asked: *"Should I say 'yes' if I am fully satisfied with it?"* P13 asked: *" If I am okay with that, but if I need some new information, should I press yes or no?"* Some participants also developed their own heuristics for giving feedback, even after the researcher's explanation. P8 described her strategy: *"One part is 'yes,' another part is 'no,' so I can say 'yes.'"*

Moreover, many participants wanted to give richer explanations to the system about why they gave specific feedback and what they want. For example, P5 gave this explanation: *"If someone is sitting and silent, maybe I want to know that he is sitting, but if someone is speaking, I don't want to know whether he is sitting or standing."*

Concerns about inconsistent feedback. Participants were concerned about providing inconsistent feedback, and took responsibility for an unsatisfactory outcome if they had given inconsistent feedback. As the training continued, participants gradually learned more about their own preferences and changed their mind about what they wanted, a form of concept evolution [12]. Thus, participants could have different responses for the same notifications in different stages of the training. Theoretically, this is not an issue for the reinforcement learning model as it will learn to adapt to users' preferences over time. However, some participants felt a need to give consistent feedback to the system. P5 expressed her own discomfort about giving inconsistent feedback: *"It is hard for me to give the feedback. At the beginning, I thought this information is good to understand this meeting. but then after training for a while, I begin to notice that some information is too much for me, I don't need to know the information. But I also thought 'oh I chose different answers before."* P12 expressed her worry that inconsistent feedback would confuse the system: *"I just made the mistake to train it to think I am happy with 'known' as knowing who that person actually is. But now if I change that, I am just gonna confuse it, it's gonna be like 'what, I thought you are happy with that?'"*

Moreover, some participants thought it was their fault if they gave inconsistent feedback. For example, P10 commented: *"I feel like I misled the system a little bit while I was still trying to figure out, during the training, what stuff I want."*

Uncertainty about the personalisation ability of the system. Participants were unsure about the capability of the reinforcement learning model, and concerned about its outcome. The capability of the reinforcement learning model refers to what preferences the system is able to learn. Even though participants were only asked to give feedback to the system and did not need to worry about the underlying model, their understating of what the system is capable of can also affect how they give feedback. For example, P6 asked the researcher: *"Maybe the second sentence came out, I will say 'yes,' but similar sentence came out at the fifth, I will just say 'no,' because at first I did not get too much information, but then I got more information, should I say 'no'?"* In this case,

P6 was satisfied with more information about the first few attendees during the overview, but wanted less information when the system moved to the following attendees because the information started to be overwhelming. However, the order of the notifications was not implemented as a feature of the reinforcement learning model, and therefore the system is not able to learn this particular preference.

Moreover, during the training, some participants wondered about how the reinforcement learning model works and the outcome of the training. For example, P2 commented that she was focused on reasoning about how the system worked rather than the notifications themselves: *"Maybe I think too much about - maybe trying to reason about what it is doing, rather focusing on the notifications themselves, and I found that distracting a little bit."* P9 asked the researcher about what will happen during the training: *"Does the training, for the whole training period, give me all the options?"* Participants were also curious about the outcome of the model. P10 asked, *"Is my score gonna converge to a certain value?"* P6 also asked the researcher: *"So what is its conclusion?"*

Post-experiment interview

Advantages and disadvantages of the two approaches. Participants identified rule-based programming as fast and straightforward, and expressed feeling more in control of the system by entering the rules themselves. Six participants in particular described liking the functionality to be able to add, edit and delete the rules. However, two participants mentioned that it is difficult to come up with the rules from scratch, while other participants had clear rules in their head in the beginning. Three participants (all non-programmers) expressed concern if they had to use the rule interface themselves without the help of the researcher. For example, P3 said: *"I would fail at the logic if I have to do that"*. However, the majority of participants believed they could learn to use the interface easily if given a tutorial or by trial and error.

With reinforcement learning, participants thought it was easy to just give simple feedback to the system, and the variance of the outputs helped them figure out what they wanted. The disadvantage was that the training process was repetitive and took a considerable amount of time.

Preference between approaches. The majority of the participants expressed preference for the rule-based programming approach over the reinforcement learning approach, due to the advantages described above. However, three participants (two of whom self-defined as non-programmers) said that they preferred the reinforcement learning approach because it learned their preferences and it was satisfying to automatically receive personalised notifications.

For this specific system and scenario, most participants thought the ideal approach would be a combination of the two approaches, while three participants (two of whom self-defined as programmers) thought they would only need rule-based programming as it was an effective way to achieve well-defined preferences, but they could see the advantages of reinforcement learning for others. For people who wanted a combination of the two approaches, all of them expressed a similar idea of being able to control the system's behaviour by entering or editing rules, while also using a pre-training phase with the reinforcement learning model to help them figure out what they want or to suggest rules. Then, the reinforcement learning model could also learn from how they use the system and suggest rules. For example, P13 mentioned that it would be helpful to detect what is a big versus a small meeting for him.

5 IMPLICATIONS FOR DESIGN

System adaptation capability & users' preferences

We found that participants were unsure about the system's adaptation capability when using both techniques. With the rule based programming, they were unsure about what the rule grammar could express. For example, do users have to define only what they want or can they specify what they do not want? For the reinforcement learning model, they were unsure about what the system could learn. For example, can the system learn that the user wants to know the pose, or that the user only wants to know when the pose is standing?

We think issue is important because the first step towards personalisation is to form preferences. In some cases, the preferences are easy to form, but there are also cases where it is difficult for users to figure out what they really want. A clear understanding of the system's personalisation capability can help them develop reasonable preferences. Rule-based programming was easy when users were sure about what they wanted and the interface supported them to achieve that. However, even when users were satisfied with the rules they created, there might not realise the system was more capable. For example, users could be satisfied with one rule for all meetings, but unaware that they could have set different rules for meetings with different sizes. The reinforcement learning approach has the advantage of exposing users to different actions that the system can take, therefore helping users figure out what they want. Then, a clear understanding of what the system is capable of learning can guide users to give feedback. A possible solution for this is to provide examples and explanations to users in the beginning.

Interface to help users express their preference

Based on the results of our study, some participants struggled to express their preferences using the rule-based programming interface, and most struggled about how to give feedback to the system when using reinforcement learning.

Rule-based programming can support users to edit/add/delete rules anytime, and therefore, the main issue for this technique centers on how to enter rules they have in mind, especially for users with little programming experience.

For the reinforcement learning model, the continuous training phase and binary feedback make it difficult for users to gradually develop their preferences and express them. For example, users cannot answer 'yes' or 'no' questions when they are not even sure what they want. Also, when participants did identify what information they wanted, there was no way for them to express their preference to avoid the lengthy process of reinforcement learning.

This suggests that systems should make clear to users what actions are needed achieve certain preferences. Possible solutions are developing interfaces to support users to give more rich feedback, for example, the feedback in the reinforcement learning scenario could be a scaled satisfaction score from 1 to 7. Or a hybrid interface could combine rule-based programming and reinforcement learning: users identify their preference through iterative feedback, but can later specify precise preferences through rules.

Help users develop cognitive model of the system

We also found out it was difficult for participants to develop a cognitive model of the personalised system. When using rule-based programming, participants became confused when dealing with multiple rules. For example, it was difficult for them to know which rules were in use in different scenarios. When using reinforcement learning, participants were concerned about the outcome of the personalised system. They did not know what the system learned, and thus the outcome of the system was unpredictable.

To help users develop their cognitive model, it is necessary to provide context. For example, if there are different rules for big and small meetings, users should be informed of the meeting size and hence which rule is in effect.

6 CONCLUSION

In this work, we studied a pure rule-based programming interface as well as a pure machine learning interface, in an identical setting, going beyond previous studies that have not compared the two alternatives in precisely the same end-user programming task. We found that for this specific system and task, the rule-based programming approach was perceived as more straightforward and easier to use than the reinforcement learning approach.

Our results suggest that an effective personalisation approach for this scenario and task would be a combination of both approaches. Reinforcement learning could help generate candidate rules, help users form preferences when they are unsure of thier preferences, detect contexts that are hard to express otherwise. Rule-based programming could help define context, explain behaviour learned by a machine learning model, and provide an interface for editing or specifying behaviour not captured by the model.

REFERENCES

[1] Louise Barkhuus and Anind Dey. 2003. Is context-aware computing taking control away from the user? Three levels of interactivity examined. In *International Conference on Ubiquitous Computing*. Springer, 149–156.

[2] AJ Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. 2011. Home automation in the wild: challenges and opportunities. In *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2115–2124.

[3] Scott Davidoff, Min Kyung Lee, Charles Yiu, John Zimmerman, and Anind K Dey. 2006. Principles of smart home control. In *International conference on ubiquitous computing*. Springer, 19–34.

[4] Anind K Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. a CAPpella: programming by demonstration of context-aware applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 33–40.

[5] Kathleen E Finn, Abigail J Sellen, and Sylvia B Wilbur. 1997. *Video-mediated communication*. L. Erlbaum Associates Inc.

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[7] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[8] Christian Heath and Paul Luff. 1991. Disembodied conduct: communication through video in a multi-media office environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 99–103.

[9] Shintami C Hidayati, Chuang-Wen You, Wen-Huang Cheng, and Kai-Lung Hua. 2018. Learning and recognition of clothing genres from full-body images. *IEEE transactions on cybernetics* 48, 5 (2018), 1647–1659.

[10] Seth Holloway and Christine Julien. 2010. The case for end-user programming of ubiquitous computing environments. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 167–172.

[11] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. 2009. Context-aware systems: A literature review and classification. *Expert Systems with applications* 36, 4 (2009), 8509–8522.

[12] Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. 2014. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3075–3084.

[13] Anastasia Kuzminykh and Sean Rintel. 2019. Let Me See: A Functional-Structural Model of Attention in Video Meetings. In *Under submission the CHI Conference on Human Factors in Computing Systems*. ACM.

[14] Paul Luff, Christian Heath, Hideaki Kuzuoka, Jon Hindmarsh, Keiichi Yamazaki, and Shinya Oyama. 2003. Fractured ecologies: creating environments for collaboration. *Human-Computer Interaction* 18, 1 (2003), 51–84.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[16] Rieks op den Akker, Dennis Hofs, Hendri Hondorp, Harm op den Akker, Job Zwiers, and Anton Nijholt. 2009. Supporting engagement and floor control in hybrid meetings. In *Cross-Modal Analysis of Speech,*

Gestures, Gaze and Facial Expressions. Springer, 276–290.
[17] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. 2017. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).
[18] Parisa Rashidi and Diane J Cook. 2009. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on systems, man, and cybernetics-part A: systems and humans* 39, 5 (2009), 949–959.
[19] Rasmus Rothe, Radu Timofte, and Luc Van Gool. 2018. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision* 126, 2-4 (2018), 144–157.
[20] Richard S Sutton, Andrew G Barto, Francis Bach, et al. 1998. *Reinforcement learning: An introduction*.
[21] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1701–1708.
[22] Anja Thieme, Cynthia L. Bennett, Cecily Morrison, Edward Cutrell, and Alex S. Taylor. 2018. "I Can Do Everything but See!" – How People with Vision Impairments Negotiate Their Abilities in Social Contexts. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 203, 14 pages. https://doi.org/10.1145/3173574.3173777
[23] Alexander Toshev and Christian Szegedy. 2014. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1653–1660.
[24] 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*. ACM, 803–812.

A ATTRIBUTES

The attributes of meeting attendees used in personalised meeting overview are described in Table 1.

B BACKGROUND AND RELATED WORK

End-user personalisation

Previous work has found rule-based programming to be a viable approach for intelligent systems, but with limitations including difficulty of learning syntax and low motivation for end-users. Machine learning approaches, such as reinforcement learning, can alleviate issues, but only for part of the personalisation problem (such as for specifying the context).

Barkhuus and Dey [1] identify three levels of interactivity for context-aware systems. Their classification makes an explicit distinction between personalisation – the user specifies the desired behaviour in each situation, passive context awareness – the system detects a change in context but leaves it to the suer to take any necessary actions, and active context awareness - the system detects changes in context and acts automatically. In a case study of various context-aware applications for a mobile device (such as changing of ringing profiles, location and activity-based alerts) the authors found that participants preferred active context awareness to personalisation, despite a perceived lack of control.

The distinction between personalisation and active context awareness as defined by this study is essentially that in both cases, the system detects changes in contexts and acts automatically. However, in the personalisation scenario, the behaviour has been defined buy the user, whereas in the active context awareness scenario, the behaviour is defined by the system designers. When the system designers correctly anticipate behaviour that end-users prefer, this works well. However, this is not an assumption we could make for the meeting scenario.

It remains an open question as to how best to combine rule-based end-user programming with machine learning and automatic inference. Many systems, such as CASAS [18] exclusively use one over the other (in this case, use machine learning over end-user programming). Moreover, context-aware computing does not have a sophisticated treatment of end-user programming. These are discussed below.

Rule-based programming in intelligent systems

Several studies have looked at rule-based programming for personalizing intelligent systems. This approach has also been referred to as trigger-action [24] and conditional logic [10]. Ur et al. [24] found that trigger-action rules can express most desired behaviours submitted by participants in an online study on smart homes. In a subsequent usability study, they found that inexperienced users can quickly learn to create programs containing multiple triggers or actions. Holloway and Julien [10] also found in an online survey that the rule-based programming paradigm (what the authors refer to as 'conditional logic') was a common method for describing desired smart home behaviours. The majority of respondents (73%) used some form of conditional logic (if, then, else, while, when) to describe the scenarios. This was the case both for programmers and non-programmers. They reported wide variation in user preferences regarding desired behaviours, as well as wide variation of approaches to achieving similar actions, demonstrating the need for a personalisable system.

Some challenges have also been identified with rule-based personalisation. Brush et al. [2] found that poor interfaces for rule-based automation led to rules that were unreliable and hard to debug. Moreover, the complexity of user desires were often a poor fit for limited rule grammars, with one participant remarking that "you can't really create hard rules to describe every single situation that you might want to automate." Ur et al. also identified a class of triggers with the term 'fuzzy triggers', which are ambiguous or person-dependent (e.g., "the water is too hot"), which the authors suggest to be an opportunity to integrate machine learning. The authors further suggest that machine learning could be

Table 1: Attributes of meeting attendees used in personalised meeting overview. The meeting simulator generated multiple meeting instances with different combinations of attendees for the user study.

Attribute	Possible Values	Description
Relation	Known, Unknown	Relation to the attendee
Name	<Individual's name>, Person 1, Person 2, etc.	Names of known attendees. For unknown attendees, their name will be Person 1, Person 2 ,etc.
Pose	Sitting, Standing, Other Pose	Other pose means any other pose except sitting and standing
Activity	Reading, Typing, Speaking, Listening, Other Activity	What the attendee is doing
Eye Gaze Mode	Other Attendee, Projector Screen, Personal Phone / Laptop, Other Object	Who or what the attendee is looking at
Location	1 to 12 o'clock	Attendee's location. We assume there is a projector screen in the meeting room. The location is clock-wise in reference to the screen at 12 o'clock
Clothing	Shirt, T-shirt, Dress	What the attendee is wearing
Gender	Male, Female	We assume users know the gender for known attendees. Therefore, gender is only provided for unknown attendees

used to resolve 'conflicts,' i.e., where different users create different rules, or where the same user creates different rules over time.

Machine learning

Machine learning from user annotated examples has been found to be effective for specifying contexts. The a CAPpella system [4] asks users to annotate captured sensor data post-hoc as being part of a certain 'situation' (e.g., a meeting), as well as pruning data from sensors deemed irrelevant. Machine learning models can then be trained to predict the context from the sensor data. This simplifies the end-user specification of a particular context by asking the user only to *identify* the context rather than *define it*.

The idea that context should be inferred from sensor data appears to be a central assumption of context-aware computing research. A survey by Hong et al. [11] characterises the process as follows: "First, algorithm is utilized to infer high-level context of user. According to levels of abstraction, context is divided into low-level context and high-level context. Low-level context is raw data collected directly from physical sensors, while high-level context is inferred from low-level context. This part involves the algorithm of context reasoner to extract high-level context and inferring algorithm to extract correct position of user, near object, and environments." When users have conflicting preferences, Hong et al. observe that a number of approaches have been applied to resolve the conflicts, including information fusion, time stamps, and fuzzy algorithms, but the problem has not been solved perfectly.

In this research, we sought to identify design guidelines for end-user personalisation of intelligent systems. As previously mentioned, context-aware programming makes limited reference to the end-user programming challenges of such systems. However, some previous work has presented design principles for smart systems, such as Davidoff et al. [3], whose principles include: "easily construct new behaviours and modify existing behaviours", and "account for multiple, overlapping and occasionally conflicting goals", which are broadly applicable.

C QUANTITATIVE RESULTS

All participants successfully completed the task. This section describes some general observations, and presents the analysis of cognitive load and satisfaction score questionnaires.

Rule-based programming

All participants were able to personalise the system by creating rules with the help of the researcher. While the rules created by the participants share some similarities (e.g., most participants wanted to know the names of all meeting attendees), participants also showed a diversity of individual preferences (e.g., some participants wanted to know the clothing information for attendees they did not know, but some participants thought the clothing information was too distracting and not useful). Seven participants defined different sets of

rules based on the properties of the meeting. For example: When the total number of attendees > 5, tell me the *name*, *location* for all attendees; When the total number of attendees < 6, tell me the *name*, *location*, *activity* for all attendees; When there is an attendee whose *activity* is speaking, tell me the *name*, *activity* for attendees whose *activity* is speaking. The other 8 participants defined a single set of rules to apply to all meetings.

Figure 4 shows the binary feedback given by each participant and the resulting training loss of the model. A feedback value of "1" means that the participant responded 'yes, I am satisfied with the notification,' and "-1" means that the participant responded 'no, I am not satisfied with the notification.' Participants encountered between 45 to 160 notifications during their respective sessions. This amount varies due to the randomized length of notifications generated by the system and participants' speed of giving feedback. We did not find any particular pattern for how participants give feedback. The training loss was calculated based on the ground truth (users' feedback) and the model's prediction on the training data. The training loss converged to values close to zero for most users, indicating that the reinforcement learning model would likely take actions consistent with the participants' feedback given during training.

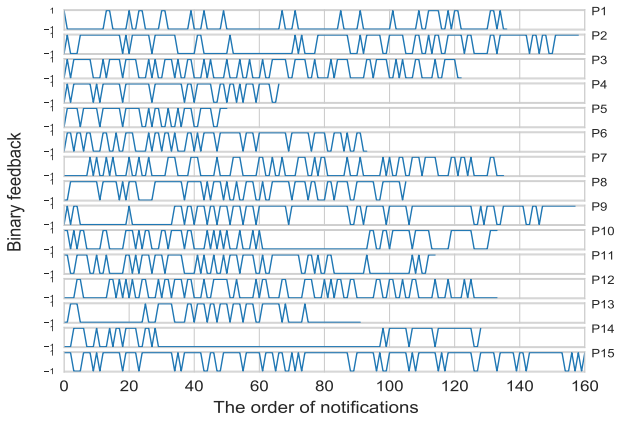
Training Cognitive Load: NASA-TLX

For the training process, participants reported a lower cognitive load when using the rule-based programming approach than the reinforcement learning approach. Based on the fact that the researcher helped the participants navigate the interface when using rule-based programming, and that the participants have to continuously give feedback to the system when using the reinforcement learning approach, it is not surprising that the participants reported a lower cognitive load when using rule-based programming. In particular, Wilcoxon Signed-rank tests show that there are significant effects on the temporal demand (the medians were 20 and 35 respectively, the TLX is within a 100-points range; z=-1.999, p=0.044, r=0.365; physical demand (the medians were 5 and 10 respectively, z=-2.120, p=0.047, r= 0.387); performance demand (the medians were 20 and 50 respectively, z=-2.684, p=0.005, r= 0.490); and frustration level (the medians were 10 and 40 respectively, z= -2.316, p=0.018, r=0.423). See Figure 5. There are no significant differences for the mental demand and effort.

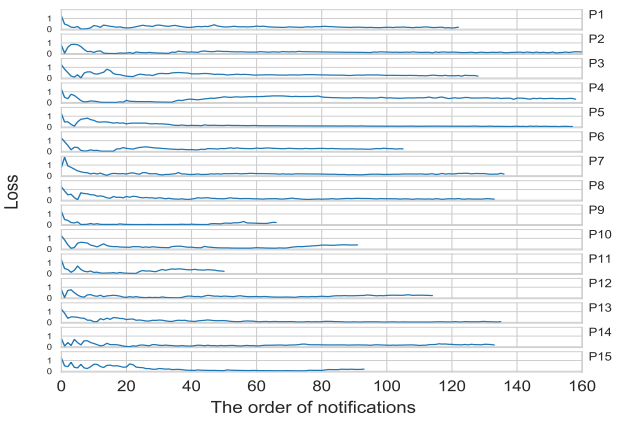
Reinforcement learning

Satisfaction with personalisation results

Wilcoxon Signed-rank tests show participants reported significantly higher satisfaction scores when using the rule-based programming approach than the reinforcement learning approach for all four questions: Q1 (the medians were 7 and 4 respectively, the satisfaction score is a 7-point scale; z=3.330, p<0.005, r=0.607); Q2 (the medians were 7 and 5 respectively, z=3.329, p<0.005, r=0.607); Q3 (the medians were 7 and 4 respectively, z=3.320, p<0.005, r=0.606); Q4 (the medians were 7 and 4 respectively, z=3.316, p<0.005, r=0.605). See Figure 6. Again, it is not surprising that participants reported a high satisfaction score when using rule-based programming. As P10 mentioned when filling out the satisfaction score for this technique: *"Of course I am satisfied, because I chose the rules myself."* This suggests that users feel more in control when using rule-based programming.



(a) Feedback given by participants: +1 and -1 correspond to is positive and negative feedback respectively.



(b) Training loss for each participant, which converged to values close to zero in most cases.

Figure 4: Reinforcement learning: binary feedback given by participants and resultant training loss

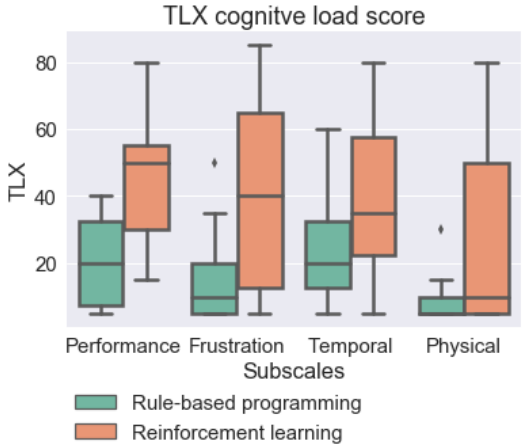


Figure 5: TLX cognitive load score for the two approaches

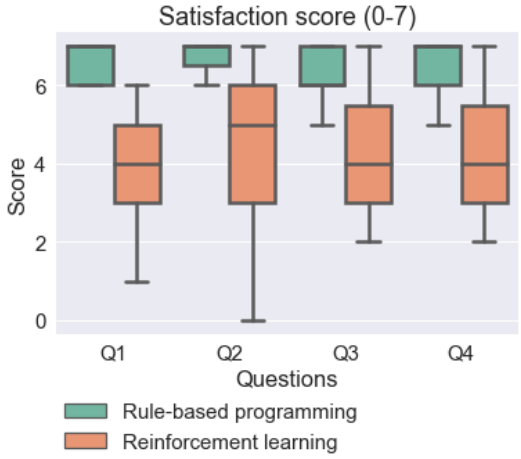


Figure 6: Satisfaction scores of the two approaches