InstructableCrowd: Creating IF-THEN Rules for Smartphones via Conversations with the Crowd

TING-HAO K. HUANG, PENNSYLVANIA STATE UNIVERSITY

AMOS AZARIA, ARIEL UNIVERSITY

OSCAR J. ROMERO, CARNEGIE MELLON UNIVERSITY
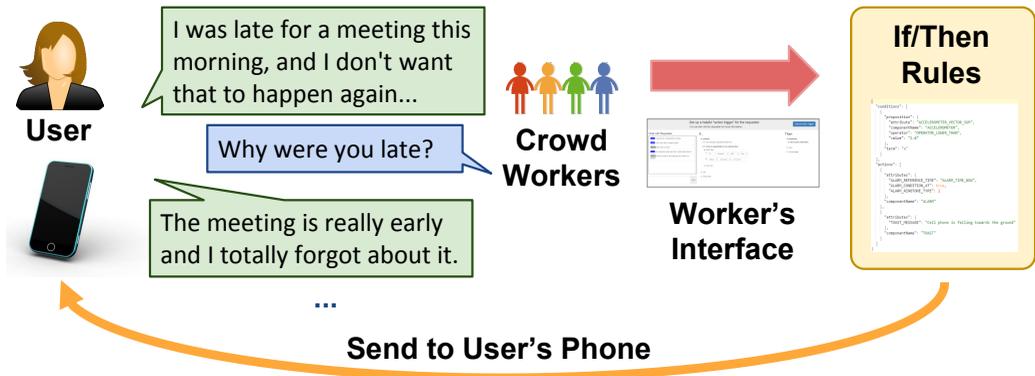
JEFFREY P. BIGHAM, CARNEGIE MELLON UNIVERSITY

ABSTRACT

Natural language interfaces have become a common part of modern digital life. Chatbots utilize text-based conversations to communicate with users; personal assistants on smartphones such as Google Assistant take direct speech commands from their users; and speech-controlled devices such as Amazon Echo use voice as their only input mode. In this paper, we introduce *InstructableCrowd*, a crowd-powered system that allows users to program their devices via conversation. The user verbally expresses a problem to the system, in which a group of crowd workers collectively respond and program relevant multi-part IF-THEN rules to help the user. The IF-THEN rules generated by InstructableCrowd connect relevant sensor combinations (*e.g.*, location, weather, device acceleration, *etc.*) to useful effectors (*e.g.*, text messages, device alarms, *etc.*). Our study showed that non-programmers can use the conversational interface of InstructableCrowd to create IF-THEN rules that have similar quality compared with the rules created manually. InstructableCrowd generally illustrates how users may converse with their devices, not only to trigger simple voice commands, but also to personalize their increasingly powerful and complicated devices.

1. INTRODUCTION

Intelligent personal computing devices – such as smartphones, smartwatches, digital assistants (*e.g.,* Amazon's Echo) and wearables (*e.g.,* Google Glass) – have become ubiquitous in society due to the power and convenience they offer. These devices are useful as shipped, but getting the most out of them requires tailoring them to their owner's preferences and needs. For example, after buying a smartphone, the user will usually first spend time customizing it by changing the wallpaper or adjusting the home screen layout. The same behavior is seen with nearly all other electronic devices, including personal assistants, tablets, laptops, and digital cameras. A great deal of customization takes place when the device is new, but the tuning process also usually continues at a slower pace

Figure 1. Users have a conversation with InstructableCrowd to create If/Then rules that then run on their phone to solve problems. The backend system is run by synchronous crowd workers who respond to the user, ask follow up questions, and create rules. Users can then review the rules on their phone to make sure they were what they wanted.

over time as users adjust their devices in response to changing needs, the availability of new software or functionality, or shifts in personal circumstances. For example, a new security threat may lead to installing better firewall software; a near-miss with severe weather may prompt the user to change local weather alert preferences; and moving to a new city may lead to changing the parameters on travel or map software to reflect the user's new location. Users manually adjust the *long-term behavior* of their devices in order to better fit their own behavior.

As important as customization is, however, it is often held back by a variety of user difficulties. One is that devices are becoming ever more complicated: new features and capabilities provide power and flexibility, but at the cost of complexity. Customizing a device often requires the user wading through complex, multi-layer menus, searching for the right app, or experimenting with poorly explained settings. All of this can be confusing and intimidating. Furthermore, getting the most from a device usually requires programming it to react intelligently to events and automate responses, and many users find programming to be difficult and even frightening. The complexity of devices also means that even a small adjustment to a system's long-term behavior through programming could result in unintended consequences to the user's experience (when compared with simple, one-time interactions such as setting an alarm). Thus, the more complex the interaction with the device, the more important are high accuracy and robustness in understanding the user's needs.

One technology has significant potential for addressing these problems is *natural language interface.* Users could much more easily customize and even automate their devices if they could simply speak to them rather than wading through instruction manuals, menu trees, and tutorials. And in fact, natural language interfaces have become a common part of modern digital life already. Chatbots utilize text-based conversations to communicate with users; personal assistants on smartphones such as Google Assistant take direct speech commands from their users; and speech-controlled devices such as Amazon Echo use voice as their only input mode.

In this exploratory project, we introduce *InstructableCrowd*, a crowd-powered system that allows

users to *program* their devices and thus change their longer-term behavior via a natural language interface. InstructableCrowd is based around two key design decisions that address the main problems with device customization and automation outlined above. First, we have focused on creating relatively simple programs that are easy to use. Second, we make use of crowd workers to operate the natural language interface instead of using automated systems, since humans are much better at understanding and interpreting complex user requirements than current electronic systems.

Our programming system is oriented around relatively simple *IF-THEN rules*, also known as *trigger-action* rules. Modern smart devices, especially smartphones, contain a wealth of *sensors* and *effectors* that can be combined to perform useful customized tasks for their users. For example, they could be used to go beyond simple, static programming (such as setting a wake-up alarm to go off at a specific time every weekday) to customizations that are based on inputs and status information (like adjusting a wake-up alarm based on traffic conditions).

A prominent example of this type of rule-based system is the mobile application IFTTT (If This Then That, ifttt.com). The service enables users to author simple trigger-action rules that contain only *one* trigger (*e.g.*, a post on Twitter) and *one* action (*e.g.*, synchronizing the latest Twitter post to Facebook) (IFTTT May 20, 2017). The service is obviously useful — it has millions of users (IFTTT May 20, 2017) — and its simplicity makes it easy to use. However, that same simplicity also means that the system fails to cover many real-world scenarios (Huang and Cakmak, 2015; Daniel et al., 2012; Ur et al., 2014). Research has shown that 22% of behaviors that people came up with require more than one sensor or effector (Ur et al., 2014). The complexity of rules people would like to create is likely to only increase as services like IFTTT continue to be integrated with other services and more devices. Therefore, in this project we focus on an extended version of IFTTT-style rules, in which the IF and THEN can each contain more than one sensor/effector.

With the awareness of the limitation of automated dialog systems, we developed a *crowd-powered* conversational agent. *InstructableCrowd* allows end users to create rich, multi-part IF-THEN rules via conversation with the crowd (Figure 1). A group of crowd workers is recruited on demand to talk with a user and create rules based on the conversation. With intelligent workers on a rich desktop interface supporting users, the interface can be simplified into a familiar speech or text chat client, allowing the system to be used on the go via mobile and wearable devices. Furthermore, users can discuss their problems with the crowd and get feedback to refine their requests. Users may know their problems, but not know what solutions would best resolve them. The crowd can help users identify possible solutions that the user didn't even know existed, and then create the rules needed to implement them. InstructableCrowd then lets users edit and improve the created rules. Controlled experiments showed that users are able to create complex rules using InstructableCrowd.

Through InstructableCrowd, we introduce a new method for enabling end users to program complex interactions with the wealth of sensors and effectors on their smartphones and other devices, which may have broader implications for the future of programming with speech.

2.   **RELATED WORK**

InstructableCrowd is related to prior work on *(i)* end-user programming, *(ii)* crowd-powered conversational agents, and (iii) automatic IF-THEN rules generation.

2.1. **End-User Programming**

InstructableCrowd builds upon the long history of research and products in end-user programming (Lieberman et al., 2006), which aims at enabling non-programmers to author or compose their own applications. Early works in this field started from database (Hanson and Widom, 1993) and email management (Mackay et al., 1989), and later gradually became more common as more and more senors and effectors became available to general users (Bolchini et al., 2007; Bronsted et al., 2010; Brush et al., 2011; Dahl and Svendsen, 2011). For instance, CoScripter allowed end-users to program scripts by demonstration (Leshed et al., 2008; Bigham et al., 2009). CoScripter used its corpus of scripts to allow easier creation of new actions from mobile devices (Lau et al., 2010); Sikuli is another famous end-user programming project (Yeh et al., 2009). Sikuli allows users to take a screenshot of a GUI element (*e.g.*, a toolbar button) and then directly use it as an element in a programming script to control the GUI's behavior (*e.g.*, clock the button.)

Trigger-action programming is one simple model of end user programming that the user form a new functionality by combining pre-defined triggers (sensors of "IF") with pre-defined actions (effectors of "THEN"). Many solutions were proposed to realize trigger-action programming, such as using existing notations of business processes modeling (BPM) to represent rules (Brambilla et al., 2012), adopting an effective workflow to create rules (Jara et al., 2013; Kokciyan et al., 2012), or solutions for domain-specific applications (Daniel et al., 2012). The IFTTT project has had great success by simplifying the composition among two applications and providing a user-friendly workflow and interface on mobile phones. The concept of IFTTT has also been extended and adopted for use in various other domain, such as smart home applications (Ur et al., 2014; De Russis and Corno, 2015), cross-device interactions (Ghiani et al., 2015), the Internet of Things (Tuomisto et al., 2014).

IFTTT only allows rules to be composed of a single trigger and a single action. Several frameworks were proposed to support multiple triggers (IFs) and actions (THENs). Dey et al. created an interface that users can drag and drop multiple sensors and effectors on a sheet to create new rules (Dey et al., 2006). Huang et al. (Huang and Cakmak, 2015) and Ur et al. (Ur et al., 2014) both extended IFTTT's interface to allow users select more than one triggers or actions. However, most of these works focused on the challenges in designing interfaces or workflows for creating a rule and examined their solutions with participants using full-size monitors and keyboards, such as via Amazon Mechanical Turk. Only few works focused on issues raised by mobile devices when creating complex rules. Häkkilä *et al.* created a trigger-action programming system, Context Studio, on the Series 60 Nokia mobile phone back in 2005 (Häkkilä et al., 2005). While the mobile devices and sensors used in Context Studio were outdated, this project provided some early insights of challenges we face today. On the other hand, competitors of IFTTT, such as Tasker, Llama, AutomateIt, On{X}, Atooma, and Microsoft's Flow all aimed to support multiple IFs and THENs in their product. However, none of these have achieved the same success as IFTTT.

Limitations of user programming were also studied. Daniel et al. (Daniel et al., 2012) pointed out that mashups platforms aimed at non-programmers are either powerful but too hard to use, or easy but too simple to be practical. Huang et al. (Huang and Cakmak, 2015) studied the mental model of IFTTT users and found that users do not always correctly understand how a sensor/effector works, which causes errors in user-created rules. Recent work has been proposed which uses crowdsourcing to build software (LaToza and van der Hoek, 2016).

2.2. **Crowd-powered Conversational Agents**

Personal intelligent agents are now available on most smartphones, *i.e.*, Google Now on Android, Siri on iOS, Cortana on Windows phones. Google Now is known for spontaneously understanding and predicting user's life pattern (*e.g.*, flight schedules, or "time to go home"), and automatically pushing notifications. Such agents can understand a number of speech commands to help users more easily access functionality. However, most of these virtual personal assistant are limited in their ability to understand their users. Google Now only reacts to certain fixed set of events, and users have no manner to extend its capability based on their own needs; Siri and Echo can perform speech queries, but are not able to understand complex verbal instructions to perform actions on the user's behalf. Although Echo allows to execute scripted actions via third-party services such as IFTTT[1], it requires users to manually program these behaviors in advance. On the other hand, InstructableCrowd gives users the direct control to define intelligent behaviors their smartphones should perform, and uses the crowd to create these behaviors with conversational interaction.

In response to this situation, crowd-powered intelligent agents were proposed. Chorus is a crowd-powered assistant that can hold intelligent conversations (Lasecki et al., 2013b) and has been deployed to public (Huang et al., 2016). Users speak to it, and it responds back quickly. Chorus is powered by a dynamic group of crowd workers (recruited on-demand) who propose responses and vote the best ones through. An incentive mechanism encourages workers to contribute useful responses. Potential downsides of crowdsourcing are cost and latency (Lasecki et al., 2013a). Guardian automates parts of Chorus by having the crowd transition existing Web APIs (Application Programming Interfaces) to dialog systems (Huang et al., 2015); and Evorus creates a framework that automates Chorus over time (Huang et al., 2018).

One limitation of these systems is that either Chorus or Guardian can only *say* something to the user, but not *do* something based on the conversation. InstructableCrowd pushes the boundaries of crowd-powered conversational systems by allowing users to perform actions beyond information inquiry. For instance, while users can discuss with Chorus to figure a good price of a flight ticket or verbally ask Guardian to query Travel APIs, users can not via a conversation configure a notification alert that monitors the dynamics of the ticket price with either systems. Enabling users to create a piece of computer-executable program via conversations opens up the opportunities of verbally "instructing" devices to customize their behaviors. The fact that today's voice-enabled devices such as Amazon's Echo allows users to set up IF-THEN rules (e.g., IFTTT) via mobile apps manually suggests the real users' needs of customizing their devices. InstructableCrowd explores performing these customization using conversational interface. A similar effort that pushes the paradigm of personal assistant toward using conversations to set up or trigger applications can also be found in recent industrial products such as Google Assistant.

Alternatively, conversational assistants powered by trained human operators such as Magic[2], Fancy Hands[3] and Facebook M have also appeared in recent years.

---

[1]Users can apply IFTTT on Amazon's Alexa manually: https://ifttt.com/amazon_alexa

[2]Magic: https://getmagic.com/

[3]Fancy Hands: https://www.fancyhands.com/

2.3.   **Automatic IF-THEN Rules Generation**

Automatically translating a natural-language utterance into the form that computers can execute is a well-known task in natural language processing, which is referred to as *language understanding* or *semantic parsing*. For instance, Artzi *et al.* used a grounded CCG (Combinatory Categorial Grammar) semantic parsing approach to map instructions such as "at the corner turn left to face the blue hall" to actions that the agent (virtual robot) can execute (Artzi and Zettlemoyer, 2013); and NaturalJava aimed to use a natural language interface for creating, modifying, and examining Java programs (Price et al., 2000).

Particularly for IFTTT rules, Quirk *et al.* collected 114,408 IF-THEN rules and their natural-language descriptions from the IFTTT website, and demonstrated the possibility of producing IF-THEN rules based on corresponding descriptive text (Quirk et al., 2015). Several follow-up work that proposed different approaches such as attention-enhanced encoder-decoder model (Dong and Lapata, 2016), using latent attention (Liu et al., 2016), or syntactic neural model (Yin and Neubig, 2017) further improved the accuracy of IFTTT rule generation. Under the context of conversational assistance, Chaurasia *et al.* created an automated dialog system that generates IFTTT rules by having a conversation with users (Chaurasia and Mooney, 2017). With a Free User-Initiative setting ("a more realistic setting"), Chaurasia's system achieved an accuracy of 81.45% in generating IFTTT rules. However, this performance is still not sufficient for practical use, and none of prior work attempted to produce multi-part rules that are more complex than that of IFTTT.

3.   **INSTRUCTABLE-CROWD**

InstructableCrowd is implemented as an Android mobile application (Figure 3) for supporting end-users to converse with crowd workers and describe problems they encounter, such as *"I was late for a meeting this morning, and I don't want that to happen again."* The crowd workers can talk with the user and use an interface to select **sensors (IFs)** and **effectors (THENs)** to create an **If-Then rule** in response to the user's problem. The rules are then sent back to the user's phone. For instance, if the user mentions having trouble with early morning meetings, the crowd can create the rule "send a notification the night before a meeting" for the user. Furthermore, InstructableCrowd is also able to merge multiple rules sent by different crowd workers to form a more reliable final rule. We describe the system architecture and implementation details in this section.

3.1.   **Rules, Sensors, and Effectors**

In this work, a **Rule** is defined as a tuple that contains an **IF** part and a **THEN** part. The IF part contains a set of **Sensors** (also referred to as **IFs**) that describe aspects of the user's life and context. For instance, the "Calendar" application describes the status of all calendar events of the user, and the "Phone Body" sensor describes the physical motions of the smart phone (*e.g.*, phone is moving). Both can be Sensors in the IF part. The THEN part contains a set of **Effectors** (also referred to as **THENs**) that can be performed, such as push a notification, set an alarm, and send a text message, etc. It is noteworthy that InstructableCrowd allows more than one Sensors/Effectors in each part, while IFTTT only allows one. An overview of an example rule is shown in Figure 2.
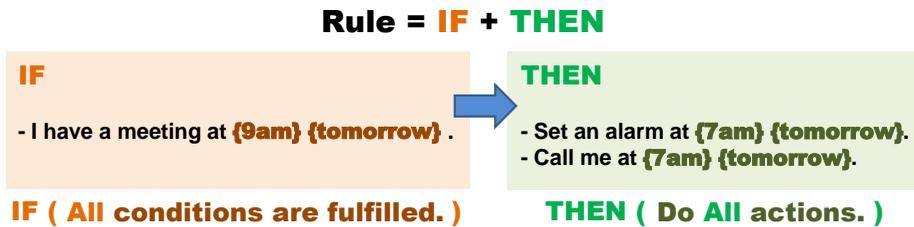
Each Sensor has one or more Triggers that can be selected. For instance, the "calendar" sensor could have three different Triggers that reflect the status of 1) currently ongoing events, 2) future events

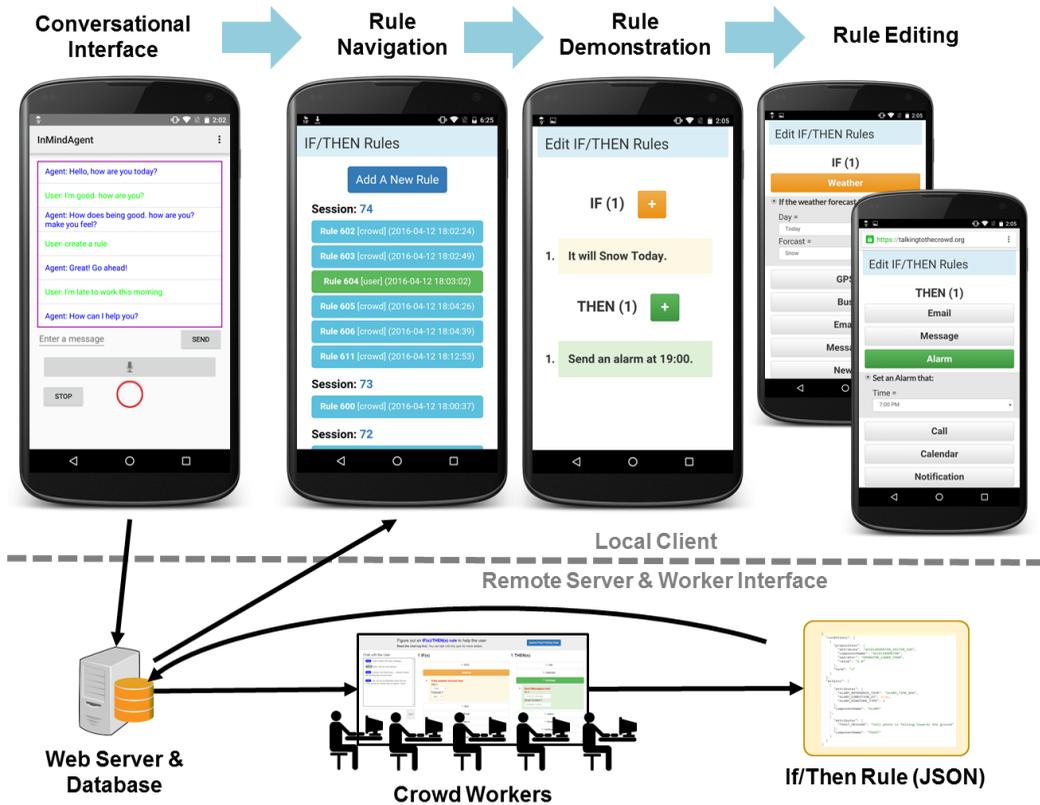| Sensor	Trigger	Trigger Description	Attributes (Input Type)
Bus	Current location	The bus is currently at a certain stop:	Bus Number (Text) Bus Stop (Text)
	Future location	The bus will arrive at a certain stop in minutes:	Bus Number (Text) Will Arrive at Stop (Text) In How Many Minutes (Text)
Calendar	Current event	If I am having an event right now that:	Event Type (Select)
	Future event (absolute time)	If I will have an event that (absolute time):	Day (Select) Start Time (Time) End Time (Time) Event Type (Select)
	Future event (relative time)	If I will have an event that (relative time):	In How Many Minutes (Text) Event Type (Select)
Call	Receive a call	If I receive a phone call that:	From (Text)
Clock	Current time	The current time is:	At/Before/After (Select) Time (Time)
Email	Receive an email	If I receive an email that:	Sent By (Text)
GPS	Current location	I am currently located at:	Location Name (Text)
	Distance to a location	If my distance to a certain location that:	To (Text) Is Greater/Less Than/Equals To (Select) Distance (Text)
Message	Receive a message	If I receive a text message that:	Sent By (Text) Contains the word(s) (Text)
News	Receive a news	If I receive a breaking news that:	Title contains the word(s) (Text)
Phone Body	Phone falls	If my phone is falling.	N/A
	Drive	If I am driving.	N/A
Weather	Weather forecast	If the weather forecast that:	Day (Select) Forecast (Select)

Table 1. Sensors (IFs) with their Triggers and Attributes as implemented in InstructableCrowd.

Effector	Action	Action Description	Attributes (Input Type)
Alarm	Set an alarm	Set an Alarm that:	Day (Select) Time (Time)
Calendar	Add an event	Add an Event on my Calendar that:	Day (Select) Start Time (Time) End Time (Time) Event Type (Text) Event Title (Text)
Call	Dial a call	Call:	To (Text) What to Say (Text)
Email	Send an email	Send Email(s) that:	To (Text) Email Title (Text) Email Content (Text)
Message	Send a message	Send Message(s) that:	To (Text) Message Content (Text)
Notification	Send a notification	Push me a Notification that:	Notification Content (Text)

Table 2. Effectors (THENs) with their Actions and Attributes implemented in InstructableCrowd.

Rule = IF + THEN

IF

- I have a meeting at {9am} {tomorrow} .

IF (All conditions are fulfilled.)

THEN

- Set an alarm at {7am} {tomorrow}.
- Call me at {7am} {tomorrow}.

THEN (Do All actions.)

Figure 2. Example of a rule in InstructableCrowd. A Rule is defined as a tuple that contains an IF part and a THEN part. The IF part contains a set of Sensors that describe aspects of the user's life and context, and the THEN part contains a set of Effectors that can be performed.

Figure 3. InstructableCrowd users have a conversation with crowd workers about a problem they are having. Crowd workers collectively create IF-THEN rules that may help the end user solve their problem using sensors and effectors available on the smartphone platform. The rules are then sent back to the user's phone for review, editing, and approval. The rules then run on the smartphone.

at an absolute time (*e.g.*, 9am today), or 3) future events at a relative time (*e.g.*, in 30 minutes.) Similarly, one Effector can also have one or more Actions to perform. Each Trigger and Action is composed of a set of **Attributes** to specify the details of the condition. For instance, for configuring "Calendar" sensor to tell if the user has any events in 30 minutes with the "Future Event (Relative Time)" Trigger, the "In How Many Minutes" attribute needs to be filled with "30," and the "Event Type" attribute needs to be filled with "Any." In this paper, we focused on observing end-user and workers behavior in **selecting Sensors/Effectors** and **filling Attributes**.

The full list of Sensor and Effectors with their Triggers/Actions and Attributes used in our study are listed in Table 1 and Table 2.

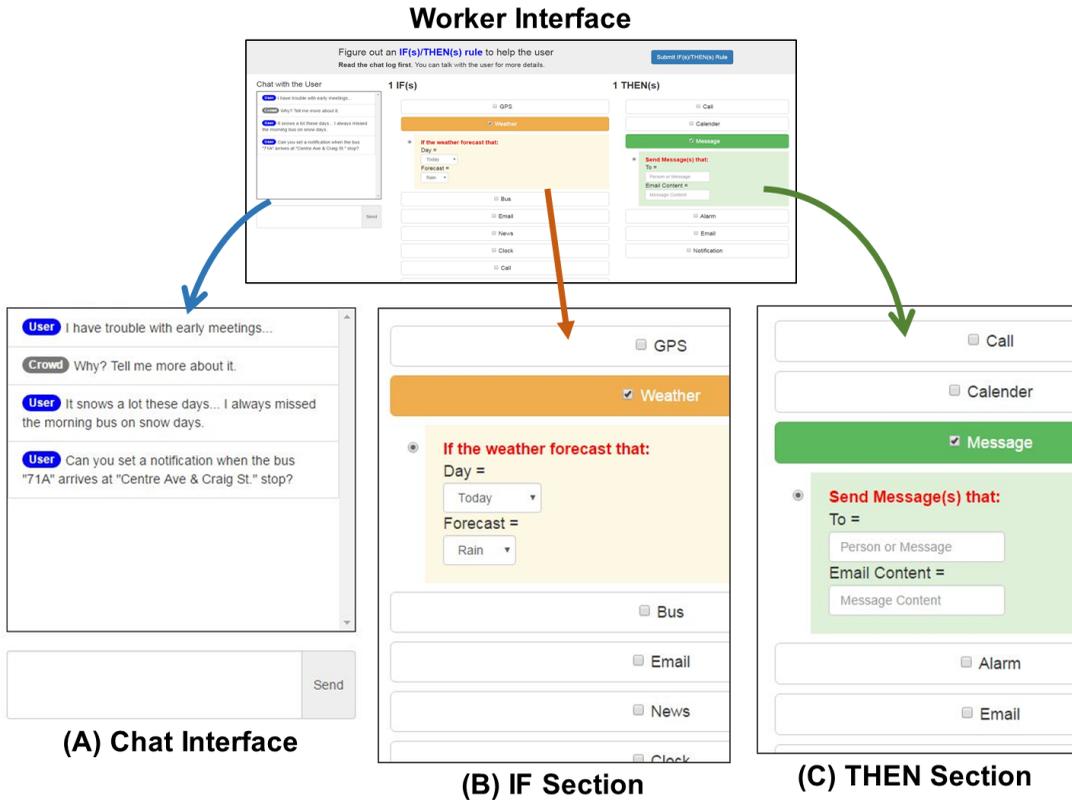## 3.2.   **Conversational Agent for the End-user**

InstructableCrowd is implemented as a conversational agent for Android smartphones. By calling the personal agent's name or clicking on the red button (as shown in Figure 3), the user is able to give the agent commands via voice or text. The client side records the user's speech and sends it to the server, which in turn sends this speech on to Google Automatic Speech Recognition; the user can also use text entry to input the command. InstructableCrowd adopts the LIA framework (Azaria et al., 2016), which uses a combinatory categorial grammar (CCG) parser to parse the input text into a logical form and execute the corresponding commands, to recognize user's voice input. Once the user give verbal commands such as *"create a rule,"* LIA connects to InstructableCrowd and initiates the rule creation process.

At the beginning of each conversation, InstructableCrowd posts 10 Human Intelligence Tasks (HITs) to Amazon Mechanical Turk to recruit a group of crowd workers. Each worker will be directed to a web-based interface (Figure 4), where they can view the user's messages, respond to the user, and compose an IF-THEN rule based on the user's request. The user and workers communicate with each other synchronously via a web server (Figure 3). A similar system framework has been used by several real-time crowd-powered conversational agents, such as Chorus (Huang et al., 2016; Lasecki et al., 2013b) and Evorus (Huang et al., 2018).

The user may then describe his problems and converse with the crowd to figure out which rules to create (the workers converse by text, and the user, may either use text or voice). Once the rule is created, it is sent back to the user's phone, where a Decision Rule Engine component (Tomazini et al., 2017; Romero and Akoju, 2018) will store, validate and process that rule. Currently, the system is implemented and tested on the Android OS 6.0.1 and the server is implemented in Java.

3.3.   **Rule Editor for the End-user**

InstructableCrowd also provides an editing interface for the user to manually create new rules, edit them and edit rules received from crowd workers. As shown in Figure 3, the user is able to navigate all received rules and click on each rule for additional details. All rules are grouped together by the conversational session in which the rule was created. Crowd-generated rules are blue, and the rules created or edited by the user are green. In order to ease on the comprehension of these rules, we created a template-based natural language description for each Trigger. For instance, the description template of "Weather" sensor's forecast Trigger is "It will [weather][day]." If "Weather" sensor's this trigger is selected, along with the "Day" attribute filled with "Tomorrow" and the "Forecast" attribute filled with "Snow", the displayed description will be "It will Snow Tomorrow." On the editing interface, the description will be generated automatically in real-time and enable the user to quickly check the rule they just created or edited. The user can also use this rule editor to manually create an IF-THEN rule from scratch on their phone without talking to the crowd. In our user study, participants use various approaches to create IF-THEN rules with InstructableCrowd. Our end-user editing interface is inspired by the IFTTT mobile APP. However, it enables the user to combine multiple IFs and THENs while IFTTT focuses on one-to-one APP compositions.

Figure 4. *Worker interface. A chat interface (A) allows workers to talk to the end user to discuss the problem. The IF section (B) allows the worker to specify Sensors, along with Triggers (in red text) and their Attributes; the THEN section (C) allows them to specify Effectors, along with Actions (in red text) and their Attributes.*

3.4. **Worker Interface**

The worker interface allows crowd workers to select Sensors (IFs) and Effectors (THENs) easily. The interface contains three main parts (Figure 4). 1) The web-based chat interface allows workers to discuss the problem with the end-user in real-time. 2) The IF section contains a set of sensors on the user's phone that describe aspects of the user's life and context. Workers first select appropriate Sensors (*e.g.*, Calendar) in the IF conditions, and then select Triggers under the Sensors (*e.g.*, Future Event (Relative Time)), and finally fill in appropriate attribute values (*e.g.*, In How Many Minutes = 30.) 3) The THEN section allows workers to select Effectors and corresponding Actions, and fill in attribute values. By selecting IFs and THENs, the worker is able to create rules that trigger certain actions based on specific conditions.

3.5. **Merge Multiple Crowd-Created Rules by Voting**

InstructableCrowd recruits multiple workers for each conversation; therefore, multiple rules are received respectively from each conversation. End users are free to pick any rules submitted by the crowd, or wait until the rules merged automatically into a *final rule*. Our automated rule-merging process uses output agreement to identify the best components to use. First, any Sensors and Effectors that are selected by more than 2 workers (our current threshold) are included in the final rule. Second, for each Sensor/Effector picked in the first step, its Trigger/Action that is selected by most workers will be chosen. Finally, for each selected Trigger/Action, InstructableCrowd fills each attribute with the value that was proposed by the most workers. If two values were proposed by an identical number of workers, InstructableCrowd selects the value which was proposed earliest. Output-agreement mechanisms such as ESP Game for collecting image labels (Von Ahn and Dabbish, 2004) have been widely used to obtain reliable human-generated labels from multiple workers (von Ahn and Dabbish, 2008). Its variation, input-agreement, has also been introduced (Law and von Ahn, 2009).

3.6. **Modular Sensors (IF) & Effectors (THEN)**

We designed a general JSON (JavaScript Object Notation) schema to represent each sensor and effector. The rules created by the crowd are represented as a combination of sensors and effectors in this JSON format. New sensors and effectors can thus be added easily. For example, the following is the Weather sensor's JSON file representing that "it will snow tomorrow" (Trigger = Weather forecast).

```
1   {
2       "name": "if-weather",
3       "condition": "if-weather-forecast",
4       "attributes": [
5           {
6               "name": "if-weather-forecast-day",
7               "value": "Tomorrow",
8               "type": "select"
9           },
10          {
11              "name": "if-weather-forecast-condition",
12              "value": "Snow",
13              "type": "select"
14          }
15      ]
16  }
```

The following is the JSON representation of the Alarm effector for "set the alarm at 7am tomorrow" (Action = Set an alarm.)

```
1   {
2       "name": "then-alarm",
```

```
3      "condition": "then-alarm-send",
4      "attributes": [
5        {
6          "name": "then-alarm-send-day",
7          "value": "tomorrow",
8          "type": "text"
9        },
10       {
11         "name": "then-alarm-send-time",
12         "value": "07:00",
13         "type": "text"
14       }
15     ]
16 }
```

The following is the JSON representation for an IF-THEN rule "IF it will snow and I have a meeting at 9am tomorrow, THEN set alarm at 7am," which includes 2 sensors (Weather and Calendar) and 1 effector (Alarm.)

```
1  {
2    "if": [
3      {
4        "name": "if-weather",
5        "condition": "if-weather-forecast",
6        "attributes": [
7          {
8            "name": "if-weather-forecast-day",
9            "value": "Tomorrow",
10           "type": "select"
11         },
12         {
13           "name": "if-weather-forecast-condition",
14           "value": "Snow",
15           "type": "select"
16         }
17       ]
18     },
19     {
20       "name": "if-calendar",
21       "condition": "if-calendar-future",
22       "attributes": [
23         {
24           "name": "if-calendar-future-day",
25           "value": "Tomorrow",
```

```
26              "type": "select"
27            },
28            {
29              "name": "if-calendar-future-type",
30              "value": "Meeting",
31              "type": "select"
32            },
33            {
34              "name": "if-calendar-future-start",
35              "value": "09:00",
36              "type": "time"
37            }
38          ]
39        }
40      ],
41      "then": [
42        {
43          "name": "then-alarm",
44          "condition": "then-alarm-send",
45          "attributes": [
46            {
47              "name": "then-alarm-send-day",
48              "value": "tomorrow",
49              "type": "text"
50            },
51            {
52              "name": "then-alarm-send-time",
53              "value": "07:00",
54              "type": "text"
55            }
56          ]
57        }
58      ]
59 }
```
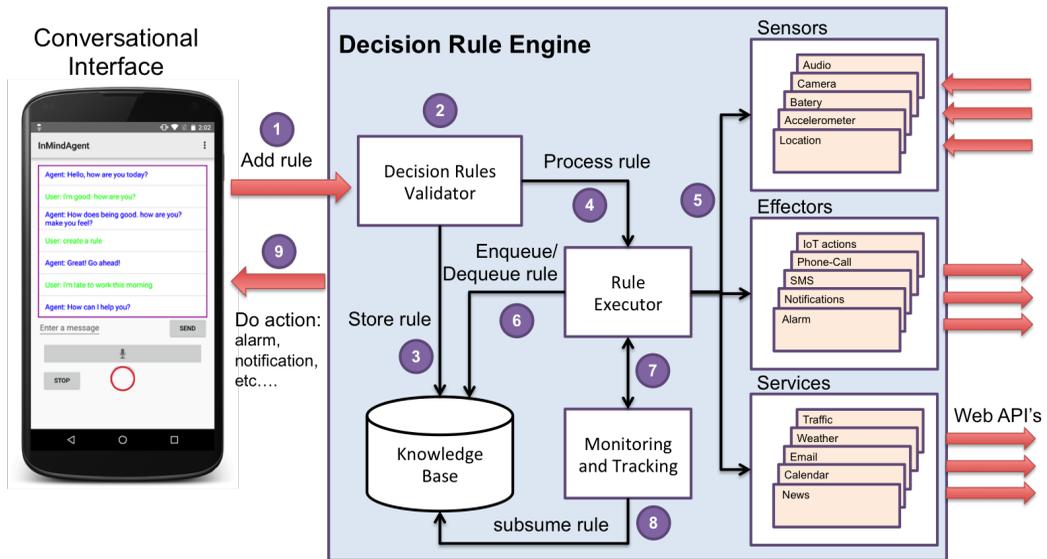
New sensors and effectors can be added easily once they are implemented in our middleware, by simply adding new JSON entries for them. Currently, we implemented 10 sensors and 6 effectors in InstructableCrowd (Table 1 and Table 2.) As we go forward, we plan to continue expand the set of available sensors/effectors.

3.7. **Decision Rule Engine**

The Decision Rule Engine is in charge of validating, storing, processing and executing rules created by either a crowd-worker or the user. Decision Rule Engine is composed of multiple modules that interact with each other in order to execute an action given a set of specific conditions that are true.

These modules are interconnected as shown in Figure 5. The following outlines the work flow (in steps), message passing and how Decision Rule Engine components cooperate over time to manage rules created by user or crowd-workers.

Figure 5. The architecture of InstructableCrowd's Decision Rule Engine. The step (1) to step (9) outlines the work flow, message passing and how Decision Rule Engine components cooperate over time to manage rules created by user and crowd-workers.

– **[Decision Rule Validator]** After the user or crowd worker has defined a new rule to be added (Step 1 in Figure 5), this component validates the syntax of that rule according to the sensors' and the effectors' attributes and constraints (Step 2). For instance, if the rule has a condition that refers to attribute <CALENDAR_START_TIME>, the validator will parse this condition and check that in fact there exists a sensor called "Calendar" that has an attribute called *startTime*, which must be of type *Date* and whose value must be a date/time that occurs later than current date/time.

– **[Knowledge Base]** Once the rule is parsed and validated, it is stored in a knowledge base where can be accessed anytime by any component (Step 3). These rules are stored locally for performance and privacy reasons, so potentially sensitive information contained within the rule is protected.

– **[Rule Executor]** After validation, the rule is immediately processed in order to determine whether it should be executed in that moment (Step 4). If so, it invokes actions from the appropriate effectors (Step 5). If not, it adds the rule to a queue so it can be executed later when all its conditions are true. The Rule Executor periodically checks to see if each enqueued rule needs to be executed (Step 6).

– **[Monitoring & Tracking]** This module is responsible for monitoring the rule execution process (Step 7) by checking if there are rules that are either never triggered or conflicting with each other (*e.g.*, one rule intends to turn the GPS on while the other one intents to turn it off.) When conflicts occur, the Monitoring/Tracking module temporarily subsumes the less relevant rule (*i.e.*, the one

that has been activated less frequently) and then user is asked to confirm this subsumption decision (Step 8).

– **[Built-in & External Sensors/Effectors]** In addition to built-in sensors and effectors that are part of the operating system, such as GPS and SMS Messages, some virtual sensors/effectors are based on external services, such as the Weather forecast and News feeds. In our implementation, we use a RESTful API to upload, extract and collect information from web servers. Finally, user is always aware of action execution through notifications, text messages, alarms, etc. (Step 9).

4. USER STUDY

For evaluating the performance of InstructableCrowd, we conducted a set of in-lab user study. Our goal is to understand if creating IF-THEN rules using conversation would sacrifice rule quality, compared with using a graphic user interface (GUI). Furthermore, we specifically recruited non-programmers because one of the benefits of using InstructableCrowd is that complex rules can be created without the need for a programming-like interface. Participants created rules using a mobile application in a control condition to allow us to compare with how users currently create rules using applications such as IFTTT.

4.1. Scenario Design

We designed the following 6 scenarios (S1 to S6) inspired by (Huang and Cakmak, 2015), along with a gold-standard set of sensors and effectors for each that we consider to be ground truth for assessing the performance.[4] We further categorized scenarios into three difficulty levels based on the numbers of sensors and effectors the scenario requires. S1 and S2 are **easy** scenarios (1 sensor and 1 effector), S3, S4, and S5 are **intermediate** scenarios (2 sensors and 1 effector), and S6 is **hard** scenario (2 sensors and 2 effectors).
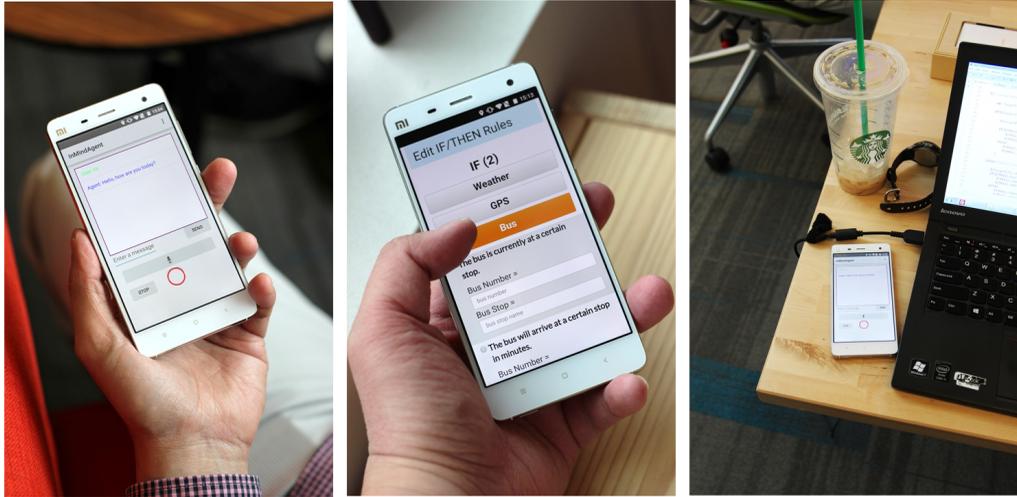
i. **[S1] Sports:** I am very interested in the performance of the "Steelers" and would like to get an immediate notification if there is a news article mentioning them. (Easy scenario.)

  – *IF: News (Receive a news: Title contains the word(s) = "Steelers")*
  – *THEN: Notification (Send a notification: Notification Content = "News of Steelers!")*

ii. **[S2] Message:** My mother likes to send me text messages. I work in a restaurant so I cannot reply to her messages very often at work. However, my grandfather was hospitalized last week and my mother is taking care of him now. I do not want to miss any important message about my grandpa. (Easy scenario.)

  – *IF: Message (Receive a message: Sent By = Mom, Contains the word(s) = "grandfather")*
  – *THEN: Notification (Send a notification: Notification Content = "Mom just texted you a message about grandfather!")*

---

[4] The attributes which were not specified in a gold-standard rule indicate that the user or worker should leave these attributes blank. In the evaluation, the textual attributes such as message content or email content will be examined manually. It is also noteworthy that in this section we only listed one common gold-standard rule, while more than one rule (*e.g.,* adding or alternating notifications) could be considered valid for a scenario. We describe the details of evaluation in Section 5.

iii.   **[S3] Snow & Meeting:** It snowed last night. I was late for work this morning and missed an important meeting at 9am because I had to take care of all the snow. My boss was quite upset and warned me this can not happen again. (Intermediate scenario.)

–   **IF**: *Weather (Weather forecast: Day = today, Forecast = snow) + Calendar (Future event [absolute time]: Day = tomorrow, Event Type = meeting, Start Time = 09:00)*
–   **THEN**: *Alarm (Set an alarm: Day = tomorrow, Time = 07:00)*

iv.   **[S4] Drive & Call:** I just heard that a large percentage of car accidents are caused by talking on the phone while driving. I decided I am not going to answer any phone calls while driving. Therefore, when I am driving, if anyone calls me, I would like to automatically reply to him/her with a message saying "Sorry I'm driving." (Intermediate scenario.)

–   **IF**: *Phone Body (Drive) + Call (Receive a call: From = Anyone)*
–   **THEN**: *Message (Send a message: To = People mentioned in "IF(s)", Message Content = "Sorry, I am driving.")*

v.   **[S5] Bus:** I usually leave work after 5pm and take Bus "53" home at the "Washington St." stop. However, the "53" buses are not common. I prefer not to wait at the bus stop unless the bus is coming soon. It takes me about 5 minutes to walk from my office to the "Washington St." stop, and it also takes about 5 minutes for Bus "53" to drive from the "Hamilton St." stop to the "Washington St." stop. (Intermediate scenario.)

–   **IF**: *Bus (Current location: Bus Number = 53, Bus Stop = "Washington St") + Clock (Current time: At/After/Before = After, Time = 17:00)*
–   **THEN**: *Notification (Send a notification: Notification Content = "Bus 53 will be arriving at Washington St. stop soon!")*

vi.   **[S6] Late for Dinner:** My wife Amy does not like me to be late home when we have a big scheduled dinner. So, if I am going to have a big dinner at home in 30 minutes, but I am still far away – say, 30 miles – from home, please send Amy a message saying "I might be home late". Also, give a phone call to "Ben's Flower Shop" and tell them to "Prepare a small surprise bouquet." (Hard scenario.)

–   **IF**: *GPS (Distance to a location: Is Greater/Less Than/Equals To = Is Greater Than, To = Home, Distance = 30) + Calendar (Future event [relative time]: Event Type = Dinning, In How Many Minutes = 30)*
–   **THEN**: *Message (Send a message: To = Amy, Message Content = "I might be home late.") + Call (Dial a call: To = Ben Flower Shop, What to Say = "Prepare a small surprise bouquet for me.")*

In our post-study survey, we asked participants to rate how realistic these scenarios are, in the scale of 1 (very unrealistic) to 7 (very realistic). The mean rating among the twelve participants was 6.25 (SD=0.62).

Figure 6. User study setting. While waiting for responses from the crowd, participants used their own laptops or mobile devices to simulate the likely context of use in the real world.

4.2. **User Study Setup**

We conducted a lab-based user study in which we asked participants to create an IF-THEN rule for each scenario using one of the following conditions:

i.   **[Condition 1] InstructableCrowd**: The participant first talks to the crowd via Instructable-Crowd (using text or voice, depends on the participant's preference) and waits to receive rules submitted from the crowd workers. The participant then selects a rule that they prefer and manually edits it to create the final rule. Each conversation was shown to 10 workers, and each worker creates an IF-THEN rule based on the conversation, respectively.

ii.   **[Condition 2] User**: The participant uses the rule editor on the phone (as shown in Figure 6) to manually create a rule.

In condition (1), three data points were recorded: the crowd-created rule that was picked by the participant (which we refer to as **Crowd Only**), the rule edited by the participant (**Crowd + User**), and the rule that was created by merging all ten crowd-created rules (**Crowd Voting**) using the process described in Section 3.5 (threshold for including a sensor/effector was 2.) We refer to condition (2) as **User Only**.

For recruiting participants, we posted the information on social media sites such as Facebook and Twitter. We also posted flyers on the campus of Carnegie Mellon University (Pittsburgh campus) and University of Pittsburgh. The goal of this project is to enable users to compose applications for their own usage, especially for the users who do not know how to program. Therefore, we recruited participants which had very limited experience in programming or none at all. People who volunteered to participate our study were directed a Web form for signing up, in which we asked

people to self-report their programming skill level ("How good are you at programming?"), from 1 ("I don't know anything about programming.") to 7 ("I'm an expert programmer."). We selected the earliest 14 participants who signed up with a self-reported programming skill level of 1 or 2. The first 2 participants were recruited for the pilot study, in which we tested and refined our study protocol and the system, and the remaining 12 participants were recruited for the formal user study. All the results reported in this paper were based on the formal user study with these 12 participants, who were aged from 26 to 36 years (mean = 29.42, SD = 3.48); 8 are female and 4 are male; and 11 participants rated their own programming skill level as 1 (out of 7), and only one participant self-rated as 2 (out of 7). It is noteworthy that the goal of this project is to examine the feasibility of using a natural language interface to create IF-THEN rules. While our participants were of a younger population, we believe that a user study with twelve participants is sufficient to show the idea of InstructableCrowd works, and that InstructableCrowd can be helpful to some users.

In our user study, we scheduled a one-hour time slot with each participant and brought them in the lab, respectively. Each participant was requested to create an IF-THEN rule which would resolve each of the 6 scenarios. The participants were asked to solve three scenarios via InstructableCrowd (condition 1), and three other scenarios via the rule editor (condition 2). The scenarios were controlled for the condition they were associated with. That is, each scenario was given to 6 subjects as condition 1 and to 6 other subjects as condition 2. In addition, the scenarios were controlled for the order in which they appeared, that is, each scenario was given in each possible order (first, second, third, fourth, fifth and last) exactly once for each condition. This was done in order to reduce the learning-effect. Participants were instructed to follow the scenarios as close as possible, but were allowed to propose minor changes during the conversation, *e.g.*, change "send me notification" to "send me an email." Participants were also free to use their own laptop or mobile devices when they waited for the response from the crowd (as shown in Figure 6,) because we believe this setting is more realistic for users who try to converse via instant messaging on mobile devices. A post-study questionnaire was used to collect subjective feedback from the participants. The compensation for each participant was $20.

For each conversational session, InstructableCrowd posted a HIT (Human Intelligence Task) with 10 assignments to MTurk. The price of each assignment was $0.50 USD. During a conversational session, multiple workers could communicate with the participant via their interface and submit rules respectively. 156 unique workers on MTurk participated in our experiments. All sessions, chats, and rules were recorded in a database with timestamps. We also timed how long the participant took to create each rule by using the rule editor.

As listed in Table 1 and Table 2, in the user study crowd workers and end-users had 10 sensors to choose from: Email, Bus, Message, GPS, Weather, Call, Clock, Calender, News, and Phone Body (for driving and phone falling); and 6 effectors: Message, Email, Alarm, Call, Notification, and Calendar (for adding an event).

5. RULE QUALITY EVALUATION

In this section we evaluated the quality of resulting rules in each setting. In order to assess the quality of a composed IF-THEN rule, we focused on two subtasks: **sensor/effector selection** and **attribute filling**. Composing an IF-THEN rule contains three sub-tasks: sensor/effector selection, trigger/action selection, and attribute filling. For instance, to effectively know that you have an early

meeting tomorrow, the "Calendar" sensor firstly needs to be selected, and then its "Future Event (Absolute Time)" trigger needs to be selected, and finally the "Start Time" attribute needs to be filled with "Before 8am." Since each sensor used in our study on average only has 1.5 triggers (SD=0.71) and each effector only has 1 action, we did not evaluate the performance of trigger/action selection separately, but merge it as a part of attribute filling. Namely, in the case that the triggers/actions selected by users or the crowd were incorrect, we noted the accuracy of attribute filling as zero in this sensor/effector.

In this section, we describe the evaluation results of InstructableCrowd and demonstrate that the system is able to produce high-quality IF-THEN rules via conversation.

5.1. **Evaluation of Sensor/Effector Selection**

The evaluation process was as follows: First, we expanded the set of our original gold-standard rules to include participant-created rules which were useful, but not exactly what we anticipated. For instance, in S3, some participants decided to send emails to the boss at work instead of setting up an earlier alarm; in S2, one participant decided to reply to his/her mom with a message instead of setting a push notification. We went through all the submitted rules and added the effective solutions that we did not think of initially. Second, we allowed extra or alternative effectors if appropriate. For instance, some participants thought that setting a push a notification is not enough and decided to send an email or to set an alarm. We considered these alternative rules are also effective. Finally, a piece of software was created to perform an automated evaluation on all recorded rules.
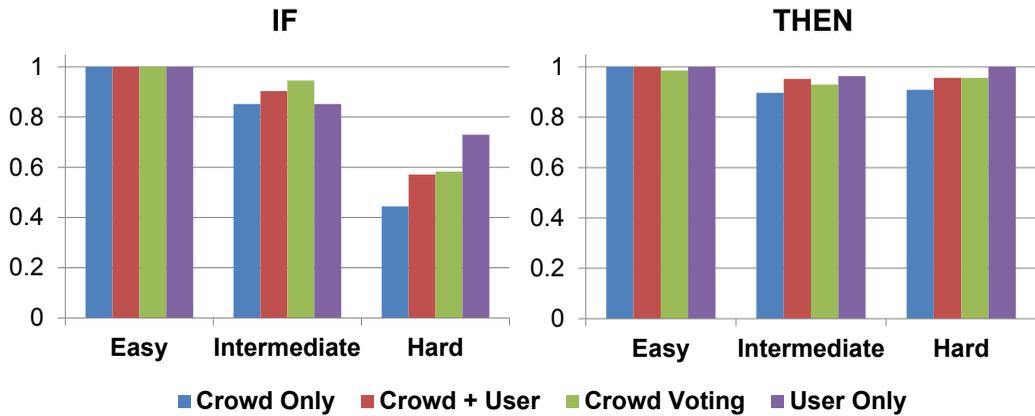
Selecting a set of correct sensors/effectors from a pool of candidate is a *retrieval* task. We therefore use precision, recall, and F1-score to evaluation this sub-task. These values are calculated as follows.

$$\text{Precision} = \frac{|\{\text{Selected Sensors}\} \cap \{\text{Gold-Standard Sensors}\}|}{|\{\text{Selected Sensors}\}|}$$

$$\text{Recall} = \frac{|\{\text{Selected Sensors}\} \cap \{\text{Gold-Standard Sensors}\}|}{|\{\text{Gold-Standard Sensors}\}|}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

When a rule is partially correct, we selected the gold-standard rule which results in the highest F1-score to report the numbers in this paper. The overall evaluation results are shown in Table 3. Both "Crowd+User" and "Crowd Voting" settings achieved comparable performances to that of the "Crowd Only" setting is both IF and THEN parts. Selecting correct sensors in IF is harder than selecting correct effectors in THEN, which is expected due to the tolerant nature of our evaluation setup for THEN. We observe that "Crowd Voting" resulted in a higher average recall, which suggested that a group of crowd workers is, collectively, less likely to forget picking some sensors than an individual user. We also notice that participants actually corrected errors in the crowd-created rules, as both the average precisions and recalls are higher in "Crowd+User" than "Crowd Only". For instance, in the "Late for Dinner" scenario (S6), one common mistake was that crowd selected only one of Calender or GPS sensors, instead of both. Two different participants fixed this error by adding back the missing sensor. Another similar example occurred in the "Bus" scenario (S5), where the

Figure 7. Average F1-score of sensor/effector selection in easy, intermediate, hard scenarios. "Crowd Voting" performed similarly or slightly better than "User Only" in easy and intermediate rules, but worse in hard rules.

crowd sometimes missed the "Clock" sensor which can indicate the current time is after 5pm. One participant fixed this by adding the Clock sensor back to the IF.

	IF			THEN			Avg
	Precision	Recall	F1 score	Precision	Recall	F1 score	F1 score
User Only	0.94	0.85	0.89	0.98	0.99	0.98	0.94
Crowd Only	0.94	0.77	0.85	0.97	0.90	0.94	0.89
Crowd+User	0.94	0.83	0.89	1.00	0.94	0.97	0.93
Crowd Voting	0.92	0.89	0.91	0.95	0.96	0.96	0.93

Table 3. Sensor/Effector selection overall performance. Both "Crowd+User" and "Crowd Voting" settings achieved comparable performances to that of the "Crowd Only" setting is both IF and THEN parts.

We also evaluated the performance based on the scenarios' difficulty level. The dynamics of F1-scores are shown in Figure 7. While the THEN parts were not influenced much, the F1-scores in IF parts' decreased as the scenarios got more complex. "Crowd Voting" performed similarly or slightly better than "User Only" in easy and intermediate rules, but worse in hard rules. These results also indicate the number of sensors and effectors influences the difficulty level of composing the rule, while other factors such as abstraction level and type of sensors/effectors also reportedly play important roles (Ur et al., 2014).

	IF	THEN	Avg
User Only	98.3%	95.0%	**96.7%**
Crowd Only	81.4%	90.0%	85.7%
Crowd + User	89.2%	93.3%	**91.3%**
Crowd Voting	86.4%	95.0%	**90.7%**

__Table 4__. Attribute filling overall performance. While the "Crowd Voting" setting achieved the same average accuracy as that of the "User Only" in the THEN part, its average accuracy is lower than "User Only" in the IF part.

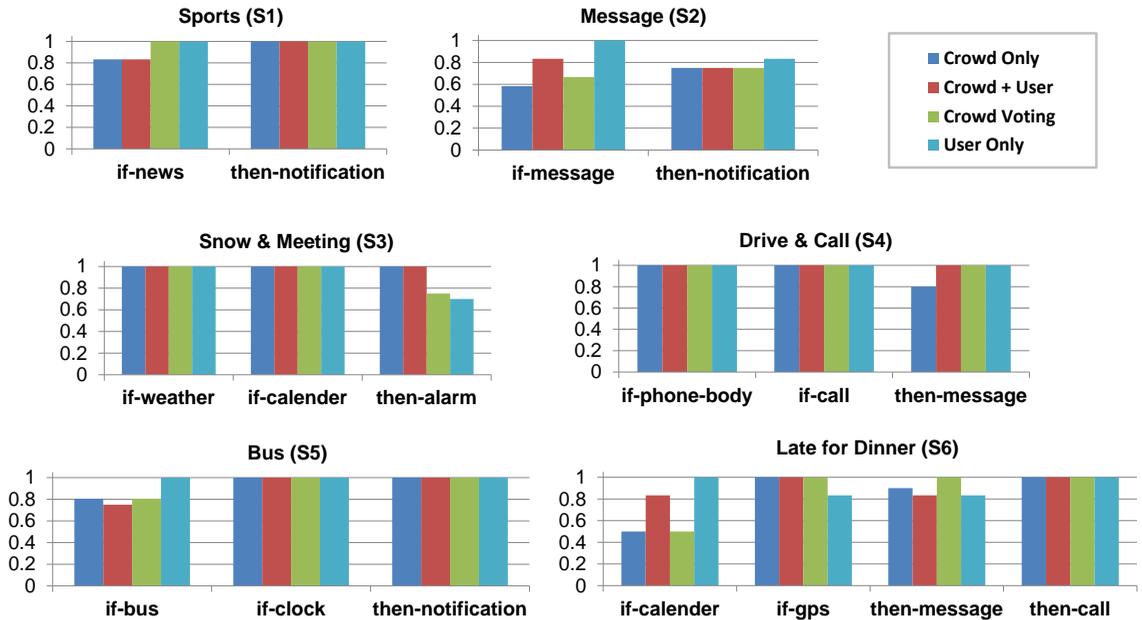## 5.2. **Evaluation of Attribute Filling**

The evaluation process of attribute filling is similar to that of sensor/effector selection. Any value for an attribute which seemed appropriate was considered to be correct. For instance, the content of the sent messages or emails could vary, and we manually labeled the effectiveness of each "content" attribute in effectors; the "Day" attribute (Table 1) in the Weather sensor of S3 could be set to either "Today" or "Tomorrow", however, it would only be judged as correct if the Alarm's "Day" attribute (Table 2) was set to the same value. Software was created to evaluate these attributes automatically.

For a given sensor/effector S that is correctly selected, we calculate the accuracy of its attribute values as follows.

$$\text{Accuracy} = \frac{\text{Number of Attributes in } S \text{ with correct values}}{\text{Number of Attributes in } S}$$

If trigger/action of S is incorrect, Accuracy = 0.

The overall evaluation results of attribute filling are shown in Table 4. While the "Crowd Voting" setting achieved the same average accuracy as that of the "User Only" in the THEN part, its average accuracy is lower than "User Only" in the IF part. To understand the sources of this performance gap, we analyzed the average accuracy of attributes in each sensor/effector of each scenario, as shown in Figure 8. We observed the sensors (IF) where "Crowd Voting" resulted in a lower accuracy than that of "User Only" (i.e., the Message sensor in S2, the Bus sensor in S5, and the Calendar sensor in S6) and identified two sources of crowd workers' errors: **communication gap** and **misunderstanding the meanings of triggers**. One source of the errors was the communication gap between the end-user and crowd workers. Namely, the user falsely expressed or missed some information when talking to the crowd. For instance, in S2, one participant falsely said "dad" often sent him/her messages (instead of "mom"), and the crowd therefore filled "dad" in the "Sent By" attribute; in S5, one participant did not mention to the crowd that it usually takes 5 minutes to walk to the bus stop, so the crowd arbitrarily filled the "In How Many Minutes" attribute of Bus sensor with 2 minutes (trigger = "Future location"). Another source of the errors is the misunderstanding the meanings of triggers. In S6, we found that some crowd workers confused the "Future Event (absolute time)" trigger with "Future Event (relative time)" trigger of the Calendar sensor. In addition, both users and crowd workers have typos in their attributes. For instance, a worker misspelled "Steelers" as "Stelers" in
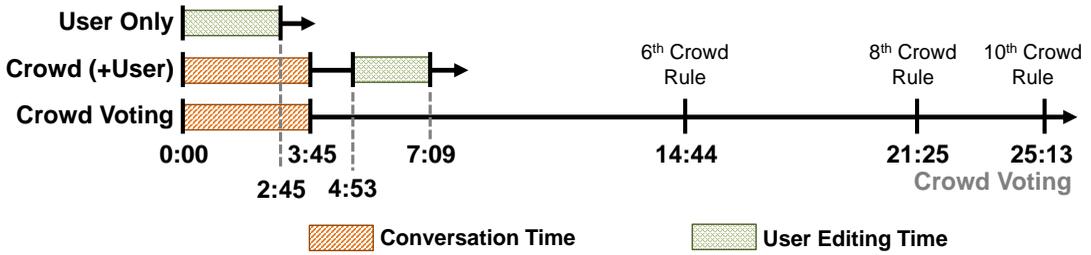
Figure 8. Average accuracy of attribute filling of correctly-selected sensors/effectors. "Crowd Voting" performed similarly as "User Only" in most cases. We analyzed S2, S5, and S6 and found that crowd errors are mainly caused by communication gap and misunderstanding of attributes.

S1, and another worker entered "19:00" as the "start time of the meeting" in S3, while the expected answer is "07:00."

6.   USER ACTIVE TIME

We also analyzed the user active time, *i.e.*, the time that **users spent on interacting with the system**. Even though it is expected that InstructableCrowd requires more time since the user needs to talk with the crowd, it is still important to understand how much time it takes a user to create a rule. In our study, participants spent an average of 2 minutes and 45 seconds (SD=1:23) to create a rule from scratch using the rule editor ("User Only"). When using InstructableCrowd, participants spent an average of 3 minutes 45 seconds (SD=2:01) to converse with the crowd, and then the system took about one minute after the conversation to create a rule that the participants were willing to pick ("Crowd Only"). If the participant decided to edit the crowd-created rules he/she just picked, it took about 2 minutes for the participants to further edit the rule ("Crowd+User"). It took approximately 20 minutes for InstructableCrowd to receive the rules from all 10 workers and calculate the final rule ("Crowd Voting"). The complete timeline is shown in Figure 9. To put these numbers in context, a study focusing on instant messaging within small groups showed that, on average (Least-Square Means), students respond to an instant message in 32 seconds, and people in startups respond in 105 seconds (Avrahami et al., 2008).

Figure 9. The complete timeline of InstructableCrowd. With the cost of a slightly longer user active time, InstructableCrowd is able to generate rules with comparable quality user-created rules. Furthermore, in our post-study survey (Section 7.1) the participants who preferred using InstructableCrowd over rule editor claimed that InstructableCrowd is "faster" or "quick", while their user active time of using InstructableCrowd is actually longer.

On average, the "Crowd Voting" setting took a user one more minute than that of the "User Only" setting. That is to say, with the cost of a slightly longer user active time, InstructableCrowd opens up a hand-free manner of creating IF-THEN rule via conversations with the crowd. We believe this is reasonable because an advantage of a speech interface is that it can be hands-free and so users can intersperse other activities while conversing to create their rules. According to our technical evaluation, the resulting rules from InstructableCrowd is as high-quality as user-created rules. It is also noteworthy that user's cognitive load when editing a rule manually and when talking with a conversational partner are very different. When having a conversation with InstructableCrowd, users are free to browse the Internet, chat with other people, or even watch a video at the same time. In our post-study survey, which we will describe in Section 7.1, the participants who preferred using InstructableCrowd over rule editor claimed that InstructableCrowd is "faster" or "quick", while their user active time of using InstructableCrowd is actually longer.

7.   QUALITATIVE RESULTS

In addition to the technical evaluation, we also collected qualitative feedback about InstructableCrowd from participants. This result suggests that InstructableCrowd provides an easier way to compose applications for the users who have difficulty creating complex rules manually on their phones.

7.1.   Feedback from Participants

We collected participants' subjective feedback immediately after they finished the lab-based study. We asked participants what method they preferred, *i.e.*, InstructableCrowd ("Crowd+User") or rule editor ("User Only"), and grouped them into two groups according to their preference. The feedback we received was that 4 participants preferred InstructableCrowd, 7 participants preferred the rule editor, and 1 participant had no preference. We also asked participants to rate the difficulty of using InstructableCrowd versus using the rule editor themselves, on a Likert scale, where 1 corresponds to very easy, 2 to easy, 3 to slightly easy, 4 to neither easy nor hard, 5 to slightly hard, 6 to hard and 7 corresponds to very hard. As shown in Table 5, compared to the participants who preferred the rule editor, the participants who preferred InstructableCrowd had a much higher difficulty rating for

using the rule editor. The correlation coefficient between a user's "difficulty rating on the rule editor" and "preferring InstructableCrowd" (prefer=1, not prefer=0) is 0.65, which is a strong correlation. Namely, the users who had a hard time using the rule editor prefer to use InstructableCrowd. A similar relation was not found between "user's difficulty rating on InstructableCrowd" and "preferring the rule editor" (correlation coefficient = 0.06). Table 5 also shows that the participants who preferred InstructableCrowd also took longer than the other group to manually compose an IF-THEN rule on average. This result suggests that InstructableCrowd provides an easier ways to create IF-THEN rules for the users who have difficulty creating complex rules manually on mobile phones. The one participant who had no preference between using the rule editor and using InstructableCrowd gave the following feedback: "it depends on different situations. for example: i would like to create rules through conversations with the system while driving." Although we recruited users without programming experience, they were somewhat tech-savvy; these results suggest we might see an even stronger effect if InstructableCrowd was used by people even less comfortable with using their smartphone.

We also asked *why* participants prefer InstructableCrowd. Interestingly, 3 out of these 4 participants said that InstructableCrowd is "faster" or "quick", while they actually spent longer time to create a rule via InstructableCrowd when comparing to the time it took them when using the rule editor. This could be because the difficult parts of creating rules is outsourced to the crowd when using InstructableCrowd, and the participants do not need to develop a rule from scratch. Some participants also stated that InstructableCrowd is more flexible since it allows the user to choose from a set of rules which is sent from multiple crowd workers. One participant who chose to use speech input said it is "faster" because he/she "doesn't like to type."

In the post-study questionnaire, we also asked participants *when* they would prefer to use Instructable-Crowd, and when they would use the rule editor. In their responses we found that people tend to create rules via conversation when 1) the rule would be too complex, and 2) they are busy or having a tight schedule. 6 out of 12 participants said they would choose InstructableCrowd when the rule they want to create has too many conditions or complex logic, e.g., "...I cannot figure out a proper logic to state 'If' and 'Then', I may relay the conversation to ask help from a server."; 3 out of 12 participants said they would choose InstructableCrowd when they are busy, e.g., "I would use it when I am busy."

7.2.   Information Inquiry, Confirmation and Suggestions in Conversations

We analyzed the conversations between the participants and the crowd, and found that the responses from the crowd were often requests for more information or explicit confirmations of user's intent. Both are known to be common dialogue acts of conversational agents (Walker and Passonneau, 2001).

Most of the conversations between users and the crowd is for collecting information. For instance, in the following conversation of S3, crowd workers ask for the information that is required in order to complete the rule they are creating:

crowd	**Hi, what can I help you with?**
user	it was snow last night and I was late for work and missed an important meeting this morning.
crowd	**Would you like a weather alert?**

		Participants Grouped by Preference	
		Prefer InstructableCrowd	**Prefer Rule Editor**
No. of participants		4	7
Avg. Difficulty Rating	**Crowd+User**	**3.25** (SD=1.50)	3.57 (SD=1.62)
	User Only	4.25 (SD=1.71)	**2.29** (SD=0.76)
Avg. Time to Create a Rule (User Only) (mm:ss)		03:15 (SD=01:20)	02:30 (SD=0:45)

Table 5. The average difficulty ratings and rule composing time of participants that prefer InstructableCrowd v.s. rule editor. Difficulty rating ranged from 1 (very easy) to 7 (very hard). The participants who preferred InstructableCrowd had a higher difficulty rating for using the rule editor, and also took longer to manually compose a rule.

crowd	**What would you like us to do?**
user	I missed an important meeting at 9am.
crowd	**What time do you usually wake up?**
user	7am
crowd	**Would you like to wake up earlier if it snows? Is 1 extra hour enough?**
user	sure.

In the following conversation of S6, a crowd worker was trying to figure out the time of the dinner:

user	if i have a big dinner on my calendar and i am going to be late (if i am still far away in 30 minutes), send my wife a message saying :" i might be home late") and call the florist to prepare a small bouquet.
crowd	**What time might this dinner start?**
user	it depends on my calendar.

In the following conversation of a different user for the same scenario S6, a different crowd worker asked similar follow-up questions:

user	I don't want to be late for home too often, otherwise my wife would get angry at me
crowd	**So how may I help you**
crowd	**when do you want to get an alert?**
user	can you send Amy a message saying I might be home late
user	yes
crowd	**what time do you want this to be sent?**
user	if I'm going to be late
crowd	**what time is late?**
user	for our scheduled dinner on my calendar

Crowd workers sometimes confirmed with the users information which was conveyed previously. For example, in the following conversation of S5, a worker asked a confirmation question about the time.

    crowd   **hello?**
      user   I leave work after 5pm and take Bus 53 home at the Washington street
      user   I don't wanna wait for the bus for too long unless the bus is coming soon
    crowd   **is after 5pm**
      user   yes

Furthermore, open conversation can lead to solutions that the user did not think of. For example, in the following conversation of S2, the crowd worker suggested to send a message back or to use an alarm/notification, instead of setting a phone call. The alternatives that the crowd came up with demonstrates their potential to be creative and think of solutions that the user might not have.

    crowd   **Hello, how can I help you??**
      user   please call me if the text from my mom containing "grandpa" or "grandfather".
    crowd   **Do you want to send them a message asking to call you, or do you want to receive an alarm or notification?**
      user   maybe just call me. thanks!

7.3. **Alternative Solutions for the Same Scenario**

We observed that participants and workers could come up with different rules in response to a same scenario, for four main reasons: First, people have their own preferred ways to be notified under different circumstances, and thus sometimes chose different effectors than we intended in their rules. For instance, more than one participant tried to add extra effectors, such as an alarm in the "Message" scenario (S2.) because they believed missing a message about the hospitalized grandfather can be quite serious. Second, similarly, users also have their own preferences for sensors. For example, in the "Snow & Meeting" scenario (S3,) one participant selected "News" in addition to the gold-standard sensors and argued that s/he would only wake up for heavy snow, which is likely to be mentioned in the news. Third, some alternative rules created by crowd workers may be caused by the ambiguities in user's instruction. For instance, in the following conversation of S4, the word "reply" does not necessarily imply "sending a message" (although it might be the most common solution). Therefore, "sending an email" is also acceptable.

      user   hi
      user   I know car accidents might happen if i talk on the phone while driving. so I would like to reply "sorry I am driving" to anyone calling me when I'm driving.
    crowd   **ok i will do so now**

Finally, sometimes two different rules can behave similarly or even identically in the real world. For example, in the "Bus" scenario (S5), the notification can either be fired when "the Bus 53 will arrive at Washington St. in 5 miniutes" or when "Bus 53 is arriving at Hamilton St. stop now," since the Bus 53 usually takes 5 minutes to drive from Hamilton St. stop to Washington St. Both rules occurred in our study.

8. DISCUSSION

In this paper we introduced InstructableCrowd, a system that allows users to create IF-THEN rules for smartphones via conversation with the crowd. This work provides a potential route toward more interesting conversations with intelligent agents than is currently possible. In this section we discuss some of the issues and reflections that came from the development and study of InstructableCrowd.

8.1. Assessing Performance and Goal Achievement

The study showed that the performance of the crowd system is nearly the same as that of a typical GUI in terms of the *quality* of the generated rules. This might lead some to question whether users would want to use InstructableCrowd if it is not *better* than other options at creating accurate IF-THEN rules. The motivation of InstructableCrowd is to challenge the traditional methods of manually composing an IF-THEN rule within the context of performing complex tasks via alternative interfaces. The key question we wanted to answer is: "Can the system perform as well as users themselves, while employing a new method of doing it?". Outsourcing complex tasks to the crowd is not always about whether or not the system can do it "better." Often it is about opening up an opportunity to achieve the same goal using a different technology or method, in this case via natural language interface. In this respect, "better" really depends on how one is assessing achievement of the goal. In prior projects within this theme, crowd-powered systems have not always performed *better* than users. In WearWrite (Nebeling et al., 2016), Chorus (Lasecki et al., 2013b), and Knowledge Accelerator (Hahn et al., 2016; Chang et al., 2016), the proposed solutions did not necessarily produce results that were faster or of higher quality than traditional methods. The value of these projects was opening up new possibilities of completing tasks in ways that were not possible before, especially with respect to flexibility. Creating a blog post by talking to a smartwatch with WearWrite will not necessarily result in higher article quality than typing it on a laptop, but the system lets users create content on the fly nearly anywhere. Searching via Chorus crowd workers might not provide better results than just using a search engine, but it is much more convenient. Similarly, Knowledge-Accelerator's use of crowd workers allows a user to ask an open-ended question and get a sophisticated answer in few hours, and open-ended questions are something that computers do not deal with very well.

8.2. Challenges in Producing High-Quality Rules

Creating a multi-part IF-THEN rule is difficult because computer-executable rules (like all programs) have little tolerance for mistakes. If we break down an IF-THEN rule to a composition of sensors and effectors with attribute values, experiments have shown that humans are reasonably good at composing sensors/effectors and filling their attributes, respectively. However, when we add up all the work, any mistakes will make the resulting IF-THEN rule ineffective. A natural response to issue would be to enforce stricter validation for human input in rule creation. However, strict input validation on the interface would increase the time it takes to create a rule for both users and the crowd and frustrate users more easily. It would also increase the engineering effort required to add a new sensor or effector to the system, which often come with arbitrary constraints. IFTTT, as a successful rule-creation product, avoids multiple sensors and effectors, and uses a user-friendly workflow to balance possible user frustration. Our project suggests using conversation and iterative editing to permit robust rule creation.

8.3. **Rule Validation**

One of the most common issues faced by the Decision-Rule Engine is the *rule conflict resolution* issue: deciding which rule should be triggered when there are multiple with the same set of conditions (IFs) but a different set of actions (THENs). If a user receives multiple rules during the same conversational session, it is reasonable to assume that they are redundant and to allow the user to pick only a single rule from this set. However, if the user creates many rules in many different sessions, he or she may forget about a created rule and attempt to create the same rule again. Furthermore, the user may at first create a very specific rule (e.g., IF I have a meeting at 9 am, THEN notify me the night before) and later try to generalize it (e.g., IF I have a meeting at 10 am or earlier, THEN notify me the night before). If the Decision-Rule Engine were to follow these rules regardless of conflicts, the same action might be executed more than once, which is not likely the user's intent. Currently, the Monitoring/Tracking module may detect these kinds of conflicts and automatically subsume the less-used rules, but further research is required into identifying these cases and alerting the user in advance. One approach could be keeping these conflicting rules and defining some heuristics that would determine when a rule should subsume or inhibit others, or when they should be executed sequentially, etc. Another approach could be defining a mechanism that removes those rules that are redundant or conflicting and less relevant than others (with the user's approval).

8.4. **Timing of Executing Triggers and Actions**

Different sensors and effectors may require very different frequencies. For example, while a weather-related sensor trigger such as "IF it is snowing early in the morning" can be checked once every 24 hours, a "phone body" sensor trigger connected to the phone's accelerometer (e.g., "IF the phone is dropping towards the floor") might need to be checked every 100 milliseconds. Other sensor conditions, such as calendar events (*e.g.* "IF I have a meeting tomorrow before 10am") may be validated immediately after the rule is created, and then checked again every hour (in case new meetings have been added). Similarly, effectors also have different execution timing requirements. Some actions can be executed immediately after the conditions are met, while others must be scheduled for later execution. For instance, the action "THEN show me a notification right now" is executed right after the conditions are fulfilled, whereas the action "THEN send me a reminder tonight at 10 pm" would be scheduled for execution at the appropriate time. Currently, the Rule Validator in InstructableCrowd's middleware uses different timing validation mechanisms for different sensors and effectors. To scale up to a larger number of sensors and effectors, a more systematic manner for categorizing the frequency ranges of sensors and effectors is likely required.

8.5. **User Privacy**

One participant in our study expressed a concern about user privacy. In the current prototype, a limited view of a user's personal information (*e.g.*, a contact list created for the purpose of the study) was exposed to crowd workers. In the future, we may use aliases that are either automatically assigned or created by the user to prevent true names or other information from being disseminated to crowd workers. For instance, instead of an actual address, the user could provide an alias such as "Home" or "Office" when talking to the crowd. Aliases can also be used to protect information about people or time (*e.g.*, using "Wife" instead of "Amy," or "Birthday" instead of the actual date.) However, the use of aliases cannot completely prevent the user from providing personal information

in conversation. While several privacy-preserving human computation workflows have been proposed for annotating videos (Lasecki et al., 2015) and accessing users' personal information (Swaminathan et al., 2017), privacy is still a well-known issue in the field of crowdsourcing, since the data is processed by human workers. A future direction is to further explore privacy issues that may arise with conversational interfaces.

8.6.   **Limitations**

One natural limitation of the architecture of InstructableCrowd is that all the sensors and effectors must be *comprehensible* to the majority of crowd workers. For example, despite being one of the most common built-in sensors in smartphones, the accelerometer sensor's raw output is very difficult to use directly by non-experts to interpret certain movements of the phone (*e.g.*, falling or being in motion while driving or walking). Future systems may find value in explicitly recruiting to their crowds people with programming expertise who can provide abstractions of raw sensor values that could be shared and reused by others. Using current sensors to express high-level semantics (*e.g.*, determining when the user is sleeping) requires specialized knowledge that most crowd workers likely do not have. IF-THEN rules have low tolerance for mistakes, and quality control is still an essential challenge in crowdsourcing. It may be useful to explore ways for the rules that are created to form a part of a probabilistic suggestion system, *i.e.*, instead of automatically conducting an action that may or may not be correct, ask the user whether or not to do it.

9.   **FUTURE WORK**

InstructableCrowd suggests a number of opportunities for future work. With the help of crowd workers, InstructableCrowd is able to convert a natural language conversation to an IF-THEN rule. Human workers are known to be able to perform various tasks that automated systems still can not do, however, often with the cost of longer latency and higher operating budget. One natural follow-up step is to explore the potential of automating the process of InstructableCrowd. While the automated approach did not perform as well as humans in prior work, a better performance can be expected when the system is able to collect larger amount of training data. Furthermore, the attribute filling task in creating IF-THEN rules is similar to the "slot filling" task in dialogue systems, in which we can take advantage of existent approaches such as Conditional Random Fields (CRF) (Raymond and Riccardi, 2007) or Recurrent Neural Networks (RNN) (Mesnil et al., 2015). Creating multi-part IF-THEN rules is a challenging task, for both human and machines. We imagine a future that automated components can work with human workers to make such systems more robust and scalable.

Furthermore, InstructableCrowd introduced a new interaction paradigm of conversational agents, which can not only be implemented in smartphones, but also be applied to smart homes, smart watches, voice-enabled devices such as Amazon's Echo, or smart cars in hand-free scenarios. End-users can freely record the problems the are experiencing and create an IF-THEN rule to solve it via any devices that are available at the spot. Figure 10 illustrates potential user scenarios of future InstructableCrowd on different devices. In a smart home setting, when the user open the smart refrigerator and find that they are out of milk, he/she can tell their Echo in the kitchen to create a rule that reminds them they do not have much milk left in the refrigerator (Figure 10 (a)); when a professor realizes that the next meeting will not be held in his/her own office but can not remember

Can you remind me every time when we are out of milk?
(a)

Remind me the locations of my meetings when they are not taking place in my office!
(b)

Ah...Okay. Send my partner a message whenever I get stuck in traffic on the way home...
(c)

Figure 10. Scenarios of future conversational assistants that allow end-users to verbally create IF-THEN rules to control smart devices. When end-users experience a problem such as (a) being out of milk, (b) forgetting the meeting room, or (c) getting stuck in traffic when driving home, they can verbally instruct their assistants at the scene to set up IF-THEN rules to prevent the problems from happening again. The framework of InstructableCrowd can not only implement on the mobile phone, but also smart watch and voice-enabled devices such as Amazon's Echo.

the room, he/she can set up a rule be talking to the smartwatch to set up a push notification about the room if the incoming meeting is in a different room; and when users get stuck in traffic when driving home, they can talk to the smart car panel and set up an automatic message whenever they will be late home (Figure 10 (c)). Voice interface opens up many possibilities of end-users to keep track of their behavior and improve life quality in the moment, and we believe that enabling users to create IF-THEN rules by talking to their smartphones is a promising start.

10. CONCLUSION

In this paper we introduced InstructableCrowd, a system that allows end users to create complex IF-THEN rules via voice in collaboration with the crowd. These rules connect to the sensors and the effectors on the user's phone where the sensors serve as triggers and the effectors as actions. We have built support for crowd workers to have a conversation with the users and allow them to suggest rules for the users. A user study shows that non-programmers can effectively create rules via conversation, and suggests that collaboration between the user and the crowd while creating IF-THEN rules could be a fruitful area for future research. As we collect examples of IF-THEN rules, we will look for ways to use them to automate the creation of common IF-THEN patterns. More broadly, InstructableCrowd represents a new approach in which end users work with remote crowd workers to bring about powerful functionality despite the constraints of mobile and wearable devices.

11.   ACKNOWLEDGEMENTS

12.   REFERENCES

Artzi, Y and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1 (2013), 49–62.

Avrahami, D, Fussell, S. R, and Hudson, S. E. (2008). IM waiting: timing and responsiveness in semi-synchronous communication. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*. ACM, 285–294.

Azaria, A, Krishnamurthy, J, and Mitchell, T. M. (2016). Instructable Intelligent Personal Agent. In *Proc. AAAI '16 (AAAI '16)*.

Bigham, J. P, Lau, T, and Nichols, J. (2009). Trailblazer: Enabling Blind Users to Blaze Trails Through the Web. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI '09)*. ACM, New York, NY, USA, 177–186. `DOI:`http://dx.doi.org/10.1145/1502650.1502677

Bolchini, C, Curino, C. A, Quintarelli, E, Schreiber, F. A, and Tanca, L. (2007). A Data-oriented Survey of Context Models. *SIGMOD Rec.* 36, 4 (Dec. 2007), 19–26. `DOI:`http://dx.doi.org/10.1145/1361348.1361353

Brambilla, M, Fraternali, P, and Vaca Ruiz, C. K. (2012). Combining social web and BPM for improving enterprise performances: the BPM4People approach to social BPM. In *Proceedings of the 21st international conference companion on World Wide Web*. ACM, 223–226.

Bronsted, J, Hansen, K. M, and Ingstrup, M. (2010). Service composition issues in pervasive computing. *IEEE Pervasive Computing* 9, 1 (2010), 62–70.

Brush, A. B, Lee, B, Mahajan, R, Agarwal, S, Saroiu, S, and Dixon, C. (2011). Home Automation in the Wild: Challenges and Opportunities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2115–2124. `DOI:`http://dx.doi.org/10.1145/1978942.1979249

Chang, J. C, Kittur, A, and Hahn, N. (2016). Alloy: Clustering with Crowds and Computation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3180–3191. `DOI:`http://dx.doi.org/10.1145/2858036.2858411

Chaurasia, S and Mooney, R. J. (2017). Dialog for Language to Code. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Vol. 2. 175–180.

Dahl, Y and Svendsen, R.-M. (2011). End-user composition interfaces for smart environments: A preliminary study of usability factors. In *International Conference of Design, User Experience, and Usability*. Springer, 118–127.

Daniel, F, Imran, M, Soi, S, Angeli, A, Wilkinson, C. R, Casati, F, and Marchese, M. (2012). Developing mashup tools for end-users: on the importance of the application domain. *Int. J. Next-Generat. Comput* 3, 2 (2012).

De Russis, L and Corno, F. (2015). HomeRules: A Tangible End-User Programming Interface for Smart Homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15)*. ACM, New York, NY, USA, 2109–2114. `DOI:`http://dx.doi.org/10.1145/2702613.2732795

Dey, A. K, Sohn, T, Streng, S, and Kodama, J. (2006). iCAP: Interactive prototyping of context-aware applications. In *International Conference on Pervasive Computing*. Springer, 254–271.

Dong, L and Lapata, M. (2016). Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280* (2016).

Ghiani, G, Manca, M, and Paternò, F. (2015). Authoring Context-dependent Cross-device User Interfaces Based on Trigger/Action Rules. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia (MUM '15)*. ACM, New York, NY, USA, 313–322. `DOI:`http://dx.doi.org/10.1145/2836041.2836073

Hahn, N, Chang, J, Kim, J. E, and Kittur, A. (2016). The Knowledge Accelerator: Big Picture Thinking in Small Pieces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2258–2270. `DOI:`http://dx.doi.org/10.1145/2858036.2858364

Häkkilä, J, Korpipää, P, Ronkainen, S, and Tuomela, U. (2005). Interaction and end-user programming with a context-aware mobile application. In *IFIP Conference on Human-Computer Interaction*. Springer, 927–937.

Hanson, E. N and Widom, J. (1993). An overview of production rules in database systems. *The Knowledge Engineering Review* 8, 02 (1993), 121–143.

Huang, J and Cakmak, M. (2015). Supporting Mental Model Accuracy in Trigger-action Programming. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. ACM, New York, NY, USA, 215–225. DOI:http://dx.doi.org/10.1145/2750858.2805830

Huang, T.-H. K, Chang, J. C, and Bigham, J. P. (2018). Evorus: A Crowd-powered Conversational Assistant Built to Automate Itself Over Time. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 295, 13 pages. DOI:http://dx.doi.org/10.1145/3173574.3173869

Huang, T.-H. K, Lasecki, W. S, Azaria, A, and Bigham, J. P. (2016). "Is there anything else I can help you with?": Challenges in Deploying an On-Demand Crowd-Powered Conversational Agent. In *Proceedings of AAAI Conference on Human Computation and Crowdsourcing 2016 (HCOMP 2016)*. AAAI.

Huang, T. K, Lasecki, W. S, and Bigham, J. P. (2015). Guardian: A Crowd-Powered Spoken Dialog System for Web APIs. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2015, November 8-11, 2015, San Diego, California.*, Elizabeth Gerber and Panos Ipeirotis (Eds.). AAAI Press, 62–71. http://www.aaai.org/ocs/index.php/HCOMP/HCOMP15/paper/view/11599

IFTTT May 20, . E. (2017). IF by IFTTT - Android Apps on Google Play. (May 2017). https://play.google.com/store/apps/details?id=com.ifttt.ifttt

Jara, J, Daniel, F, Casati, F, and Marchese, M. (2013). From a simple flow to social applications. In *Current Trends in Web Engineering*. Springer, 39–50.

Kokciyan, N, Uskudarli, S, and Dinesh, T. (2012). User generated human computation applications. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, 593–598.

Lasecki, W. S, Gordon, M, Leung, W, Lim, E, Bigham, J. P, and Dow, S. P. (2015). Exploring Privacy and Accuracy Trade-Offs in Crowdsourced Behavioral Video Coding. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1945–1954.

Lasecki, W. S, Thiha, P, Zhong, Y, Brady, E, and Bigham, J. P. (2013)a. Answering Visual Questions with Conversational Crowd Assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. ACM, New York, NY, USA, Article 18, 8 pages. DOI:http://dx.doi.org/10.1145/2513383.2517033

Lasecki, W. S, Wesley, R, Nichols, J, Kulkarni, A, Allen, J. F, and Bigham, J. P. (2013)b. Chorus: A Crowd-powered Conversational Assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 151–162. DOI:http://dx.doi.org/10.1145/2501988.2502057

LaToza, T. D and van der Hoek, A. (2016). Crowdsourcing in Software Engineering: Models, Motivations, and Challenges. *IEEE Software* 33, 1 (2016), 74–80.

Lau, T, Cerruti, J, Manzato, G, Bengualid, M, Bigham, J. P, and Nichols, J. (2010). A Conversational Interface to Web Automation. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 229–238. DOI:http://dx.doi.org/10.1145/1866029.1866067

Law, E and von Ahn, L. (2009). Input-agreement: A New Mechanism for Collecting Data Using Human Computation Games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1197–1206. DOI:http://dx.doi.org/10.1145/1518701.1518881

Leshed, G, Haber, E. M, Matthews, T, and Lau, T. (2008). CoScripter: Automating & Sharing How-to Knowledge in the Enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1719–1728. DOI:http://dx.doi.org/10.1145/1357054.1357323

Lieberman, H, Paternò, F, Klann, M, and Wulf, V. (2006). *End-user development: An emerging paradigm*. Springer.

Liu, C, Chen, X, Shin, E. C, Chen, M, and Song, D. (2016). Latent attention for if-then program synthesis. In *Advances in Neural Information Processing Systems*. 4574–4582.

Mackay, W. E, Malone, T. W, Crowston, K, Rao, R, Rosenblitt, D, and Card, S. K. (1989). How do experienced Information Lens users use rules? 20, SI (1989).

Mesnil, G, Dauphin, Y, Yao, K, Bengio, Y, Deng, L, Hakkani-Tur, D, He, X, Heck, L, Tur, G, Yu, D, and others, . (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 3 (2015), 530–539.

Nebeling, M, To, A, Guo, A, de Freitas, A. A, Teevan, J, Dow, S. P, and Bigham, J. P. (2016). WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3834–3846. DOI:http://dx.doi.org/10.1145/2858036.2858169

Price, D, Rilofff, E, Zachary, J, and Harvey, B. (2000). NaturalJava: a natural language interface for programming in Java. In *Proceedings of the 5th international conference on Intelligent user interfaces*. ACM, 207–211.

Quirk, C, Mooney, R. J, and Galley, M. (2015). Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes.. In *ACL (1)*. 878–888.

Raymond, C and Riccardi, G. (2007). Generative and discriminative algorithms for spoken language understanding.. In *INTERSPEECH*. 1605–1608.

Romero, O. J and Akoju, S. (2018). An Efficient Mobile-Based Middleware Architecture for Building Robust, High-Performance Apps. In *Proceedings of the IEEE International Conference on Software Architecture Companion (ICSA-C)*. 97–100.

Swaminathan, S, Fok, R, Chen, F, Huang, T.-H. K, Lin, I, Jadvani, R, Lasecki, W, and Bigham, J. (2017). WearMail: On-the-Go Access to Information in Your Email with a Privacy-Preserving Human Computation Workflow. In *30th ACM Symposium on User Interface Software and Technology (UIST 2017)*.

Tomazini, L, Romero, O. J, and Hruschka, E. J. (2017). An Architectural Approach for Developing Intelligent Personal Assistants Supported by NELL. In *ENIAC (Encontro Nacional de InteligÃłncia Artificial e Computacional)*.

Tuomisto, T, Kymäläinen, T, Plomp, J, Haapasalo, A, and Hakala, K. (2014). Simple Rule Editor for the Internet of Things. In *Intelligent Environments (IE), 2014 International Conference on*. IEEE, 384–387.

Ur, B, McManus, E, Pak Yong Ho, M, and Littman, M. L. (2014). Practical Trigger-action Programming in the Smart Home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 803–812. DOI:http://dx.doi.org/10.1145/2556288.2557420

Von Ahn, L and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 319–326.

von Ahn, L and Dabbish, L. (2008). Designing Games with a Purpose. *Commun. ACM* 51, 8 (Aug. 2008), 58–67. DOI:http://dx.doi.org/10.1145/1378704.1378719

Walker, M and Passonneau, R. (2001). DATE: a dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Proceedings of the first international conference on Human language technology research*. Association for Computational Linguistics, 1–8.

Yeh, T, Chang, T.-H, and Miller, R. C. (2009). Sikuli: Using GUI Screenshots for Search and Automation. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 183–192. DOI:http://dx.doi.org/10.1145/1622176.1622213

Yin, P and Neubig, G. (2017). A Syntactic Neural Model for General-Purpose Code Generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada. https://arxiv.org/abs/1704.01696