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

Ting-Hao (Kenneth) Huang
Carnegie Mellon University
Pittsburgh, PA 15213, USA
tinghaoh@cs.cmu.edu

Amos Azaria
Carnegie Mellon University
Pittsburgh, PA 15213, USA
azariaa@cs.cmu.edu

Jeffrey P. Bigham
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jbigham@cs.cmu.edu

CHI'16 Extended Abstracts, May 7–12, 2016, San Jose, CA, USA.
ACM 978-1-4503-4082-3/16/05.
http://dx.doi.org/10.1145/2851581.2892502

Abstract
In this paper, we introduce InstructableCrowd, a system that allows end-users to instruct the crowd to create trigger-action ("if, then") rules based on their needs. We create a framework which enables users to converse with the crowd using their phone and describe a problem which they might have. We create an interface for a crowd worker to both chat with the user and compose a rule with an "IF" part connected to the user's phone sensors (e.g. incoming emails, GPS location, meeting calendar, weather information etc.), and a "THEN" part connected to user's phone effectors (e.g. sending an email, creating an alarm, posting a tweet, etc.). The system then sends the rules created by the crowd to the user's phone in order to help the user solve his problem.

Author Keywords
Crowdsourcing; Crowd-Powered System; Mobile; End-User Programming

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction

In this project, we propose a new approach to utilize crowdsourcing to enable smart phone users to create their own applications. There has been a long history of interest of *end-user development* and *mashup* technologies, which aim to enable end-users without professional programming skills to create applications based on their own needs. One of the most well-known projects in this field is the IFTTT (If This Then That)[1]. IFTTT enables end-users to author simple Event-Condition-Action (ECA) rules which contain triggers (e.g., a post on Twitter) and actions (e.g., synchronize the latest Twitter post to Facebook). However, as pointed out by Daniel et al. [4], most general-purpose application composition solutions aimed on end-users have some limitations: They either expose too much functionality and become too challenging for non-programmers, or over-simplify the compositionality and become too simple for practical use. These systems also require that the users know exactly what their ECA would be. A user having a problem which he is trying to solve, but not knowing what the ECA for solving it is, or having only a vague understanding of what he wants may not find these systems useful. These systems also do not support natural language and speech commands which may be the main form of communications for some smart devices such as smart-watchs and smart-glasses.

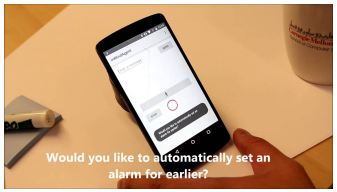

Figure 1: InstructableCrowd personal agent implemented on an Android phone.

In this work, we introduce **InstructableCrowd**, a crowdsourcing approach that enables users to create IF/Then rules. We design a system framework (Figure 2) and interface (Figure 1) that allows smart

[1] IFTTT: https://ifttt.com/

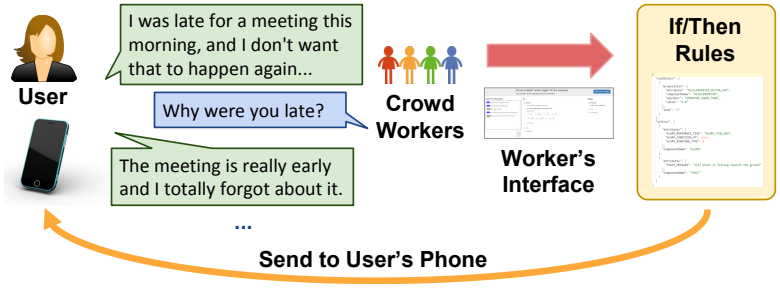

Figure 2: InstructableCrowd system overview.

phone users to instruct crowd workers to compose an application for them via conversation. By outsourcing the cognitive load of understanding the detailed functionality supported by the smart phone, end-users are able to focus on the problems they want to solve. With a conversational interface, users can also discuss their problems with the crowd and thus get instant feedback to refine their requests. Note that the user may initially only know what his problem is but not what the solution might be. The crowd can help identify a possible solution with the user and then also realize it for him.

The crowd workers can provide several advantages for the users: First, crowd workers perform the task on full-screen computers with keyboards and mouses rather than mobile devices. Second, workers are more familiar with the operations that can be performed on the phone. Each worker must read instructions and might interact with several different users during each task. Third, we assume that most crowd workers have slightly more experience in using computers

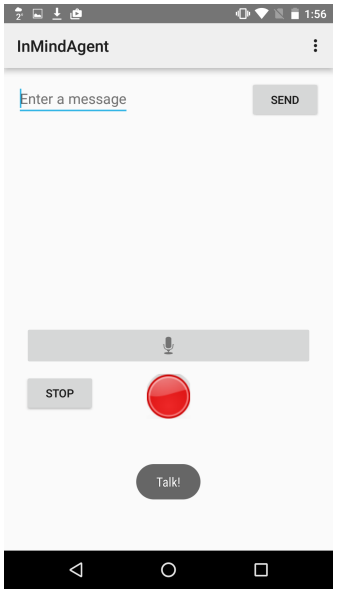

Figure 3: InstructableCrowd's end user interface.

than average smart phone users. Finally, crowd workers can be viewed as an additional cognitive resource that provide "wisdom of the crowd" effects.

Related Work

User Development & Mashups There have been many projects developed to enable non-programmer users to author or compose their own applications. Daniel et al. [4] pointed out the limitations of most of existing user development or mashups platforms aimed on non-programmers: they are either powerful but too hard to use, or easy but too simple to be practical. In response to this situation, several solutions have been proposed.

CoScripter allowed end users to program scripts to act on their behalf by demonstration [13, 2], and used its corpus of scripts to make creating new actions easier from mobile devices [12]. The IFTTT project reached to its great success by simplifying the composition among two applications and providing a user-friendly workflow and interface on mobile phones. The Atooma[2] project is similar to IFTTT but supports multiple applications as triggers and actions. The concept of IFTTT is also extended for use in smart home applications [14, 5] and cross-device [6] rules. Some projects extend existing business processes modeling (BPM) notations to simplify the composition of applications [3]; Some works focus on domain-specific applications instead of general purpose platforms [4], and others focus on designing a more effective workflow of creating rules [8, 9].

Crowd-powered Conversational Agents Chorus is a crowd-powered assistant that can hold intelligent conversations about almost anything [11]. End-users speak to it, and 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 [10]. Automating parts of Chorus by having the crowd transition existing Web APIs (Application Programming Interfaces) to dialog systems can make it cheaper [7]; Alternatively, conversational assistants powered by trained human operators such as Magic[3] and Facebook M have also appeared in recent years.

InstructableCrowd System

The system overview of the **InstructableCrowd** is shown in Figure 2. The end-user is able to converse with crowd workers to describe the problems he encountered, 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 example, 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; if the user wants extra time in the morning to clean up the car if it snowed during the night, the crowd can create a rule to set the alarm 10 minutes earlier than usual.

[2] Atooma: https://www.atooma.com/

[3] Magic: http://getmagicnow.com/

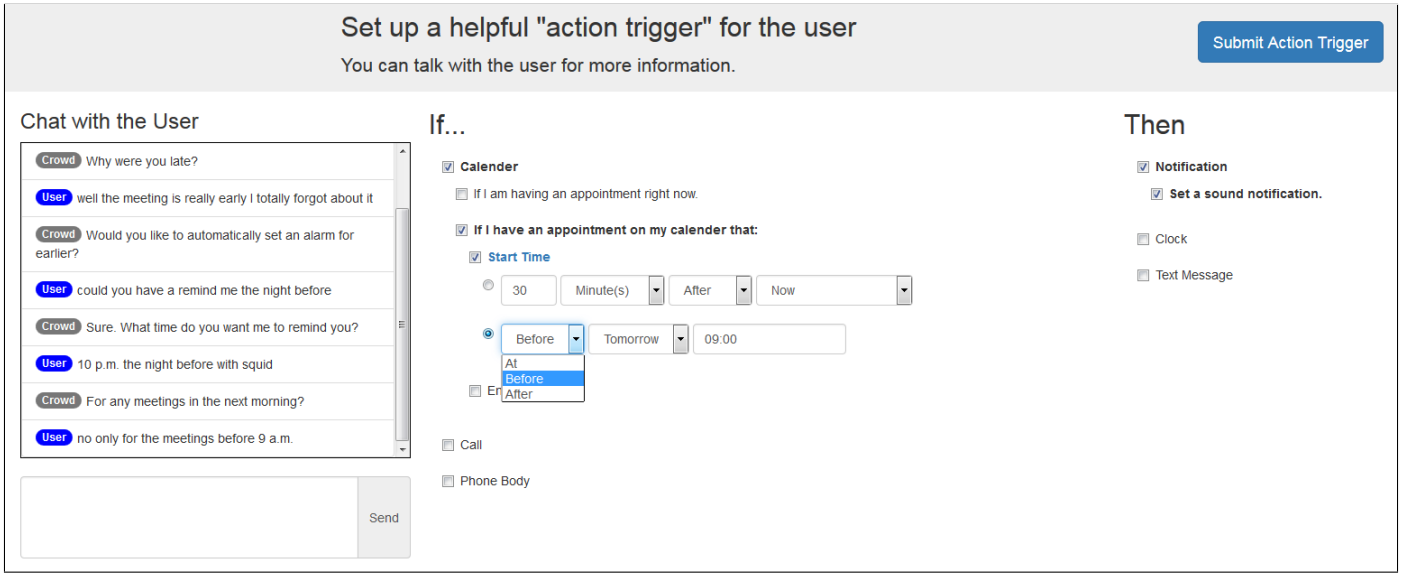

Figure 4: Worker interface. A chat interface (left) allows workers to talk to the end user to discuss the problem. The "IF" section (middle) allows the worker to specify conditions and the "Then" (right) allows them to specify effectors.

Conversational Agent for the End-user The InstructableCrowd is implemented as a conversational agent on the smart phones (Figure 1). The detailed end-user's interface is shown in Figure 3. By calling the personal agent's name or clicking on the red button, the user is able to give the agent voice commands. The client side records user's speech and sends it to the server, which runs Google Automatic Speech Recognition. Once the agent receives the command as text, we use the framework described in [1], named LIA (Learning by Instruction Agent), to execute the command. LIA uses a combinatory categorial grammar (CCG) parser to parse the input text into a logical form and execute the corresponding commands. In the InstructableCrowd system, we recognize verbal commands such as "Create a rule" to initiate the rule creation process. The end-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, the rule is sent back to the user's phone and applied by the middleware component running on the phone. Currently, the system is implemented and tested on the Android OS 6.0.1., with the server end implemented in Java.

Worker Interface We create an interface for crowd workers to select IFs and THENs easily. The interface

contains 3 main parts (Figure 4). The web-based chat interface allows workers to discuss the problem with the end-user in real-time. The "IF" section contains a set of **sensors** on the user's phone that describe aspects of the user's life and context. For instance, the Google Calender 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 are considered "sensors" in InstructableCrowd. Workers select appropriate trigger sensors in the "IF" conditions. The "Then" section allows them to select corresponding **effectors**. Effectors are the actions that can be performed on user's smart phone such as push a notification, set an alarm, and send a text message, etc. By selecting "IFs" and "THENs", the worker is able to create rules that trigger certain action based on specific conditions.

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 sensor and effector JSON. New sensors and effectors can thus be added easily.

For example, the following is the Google Calender sensor's JSON file representing that *"a calendar event will start at 9:30 tomorrow"*.

```
{
    "proposition": {
        "attribute": "CALENDAR_START_TIME",
        "componentName": "CALENDAR",
        "operator": "OPERATOR_TIME_EQUAL",
        "value": "9:30",
        "referenceAttribute": "
            CALENDAR_TOMORROW"
    }
}
```

The following is the JSON representation of the effector that *"ring the alarm now"*.

```
{
    "attributes": {
        "ALARM_REFERENCE_TIME": "
            ALARM_TIME_NOW",
        "ALARM_CONDITION_AT": true,
        "ALARM_RINGTONE_TYPE": 2,
        "ACTION_TYPE": "ALARM"
      },
      "componentName": "ALARM"
}
```

The following is the general JSON representation for a trigger-action rule, which includes a list of sensor (IFs/condition) items and effector (action) items.

```
{
    "actions": [
        {action_1}, ... , {action_n}
    ],
    "conditions": [
        {condition_1}, ..., {condition_n}
    ],
    "ruleID": "rule_1"
}
```

These rules and actions are made possible by a general architecture that we have built to allow systems to access a list of sensors and effectors, and then specify what sensor conditions should lead to what actions. These are modular, allowing new sensors and effectors to be added easily. As we go forward, we will continue to expand the set of available sensors and actions.

Pilot Study on Amazon Mechanical Turk

To examine crowd workers' capability of discussion with end-users and composing a rule for them by selecting appropriate sensors and effectors, we conducted a pilot study on Amazon Mechanical Turk (MTurk). We followed the process used to test Chorus [11]: Prior to the experiment we generated a script and a task. An experiment manager took the role of a user and followed the script as closely as possible during the experiment, while still allowing the conversation to flow naturally. The task we designed for the experiment is to ask the crowd to *"set both an alarm and a notification if the user has an early meeting tomorrow."* 5 workers were recruited on MTurk to accomplish this task, interacting via the worker's interface (Figure 4). In our study, all of the workers selected the correct sensors and effectors, and constructed the correct rule. The following is an example of a conversation between the user and the worker.

- **user:** I'd like to set up an alarm
- **user:** to remind me if i have early meeting the next morning
- **crowd: What time do the meetings usually take place?**
- **user:** remind me if the meeting is before 9am next day
- **user:** set an alarm
- **user:** and both email
- **user:** to <email_address>
- **crowd: You got it!**

Conclusion and Future Work

In this work, we introduced InstructableCrowd, a system that allows a user to instruct the crowd to create a If-Then rule via conversation. This rule connects the sensors and effectors which are active on the user's phone. We have created within the middleware a generic JSON representation for available sensors and effectors, along with the ability to define rules. We have built support for crowd workers to have a conversation with the users and define these rules for them. Our continued work will focus on enabling robust creation of IF-THEN rules via voice. 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. Users will want feedback when collaborating with the systems to have confidence that what they intended to create was actually created. We will work with users to develop appropriate ways for them to validate and edit the rules that are created.

Acknowledgements

This work was supported by Yahoo! InMind and National Science Foundation Award #IIS-1149709. We thank our anonymous reviewers for their comments.

REFERENCES

1. Amos Azaria, Jayant Krishnamurthy, and Tom M Mitchell. 2016. Instructable Intelligent Personal Agent *(AAAI '16)*.

2. Jeffrey P. Bigham, Tessa Lau, and Jeffrey Nichols. 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**

3. Marco Brambilla, Piero Fraternali, and Carmen Karina Vaca Ruiz. 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.

4. Florian Daniel, Muhammad Imran, Stefano Soi, AD Angeli, Christopher R Wilkinson, Fabio Casati, and Maurizio Marchese. 2012. Developing mashup tools for end-users: on the importance of the application domain. *Int. J. Next-Generat. Comput* 3, 2 (2012).

5. Luigi De Russis and Fulvio Corno. 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**

6. Giuseppe Ghiani, Marco Manca, and Fabio Paternò. 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**

7. Ting-Hao (Kenneth) Huang, Walter S. Lasecki, and Jeffrey P. Bigham. 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**

8. Juan Jara, Florian Daniel, Fabio Casati, and Maurizio Marchese. 2013. From a simple flow to social applications. In *Current Trends in Web Engineering*. Springer, 39–50.

9. Nadin Kokciyan, Suzan Uskudarli, and TB Dinesh. 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.

10. Walter S. Lasecki, Phyo Thiha, Yu Zhong, Erin Brady, and Jeffrey P. Bigham. 2013a. 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**

11. Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. 2013b. 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**

12. Tessa Lau, Julian Cerruti, Guillermo Manzato, Mateo Bengualid, Jeffrey P. Bigham, and Jeffrey Nichols. 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**

13. Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 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**

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