Toward Truly Personal Chatbots

On the Development of Custom Conversational Assistants

Florian Daniel
Politecnico di Milano, DEIB
Milan, Italy
florian.daniel@polimi.it

Maristella Matera
Politecnico di Milano, DEIB
Milan, Italy
maristella.matera@polimi.it

Vittorio Zaccaria
Politecnico di Milano, DEIB
Milan, Italy
vittorio.zaccaria@polimi.it

Alessandro Dell'Orto
Politecnico di Milano, DEIB
Milan, Italy
alessandro2.dellorto@mail.polimi.it

ABSTRACT

Chatbots, i.e., conversational software agents able to interact with users via instant messaging channels like Messenger, WhatsApp or SMS, have the power to substantively simplify human-computer interaction thanks to their natural language paradigm. While this certainly helps to lower barriers, state-of-the-art chatbots prevalently provide access to generic, non-personalized features with relatively little usefulness. This may hinder adoption. To provide users with real value, we envision a kind of chatbot that is personal and helpful by providing services that are chosen and configured by the users themselves, for themselves. As the development of a one-size-fits-all, yet flexible and customizable bot is hard, if not impossible, we discuss requirements and design options that directly put the user into control of their own personal bot.

KEYWORDS

Personal chatbots, platform, development, end-users

ACM Reference Format:
Florian Daniel, Maristella Matera, Vittorio Zaccaria, and Alessandro Dell'Orto. 2018. Toward Truly Personal Chatbots: On the Development of Custom Conversational Assistants. In *SE4COG'18: SE4COG'18:IEEE/ACM 1st International Workshop on Software Engineering for Cognitive Services , May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3195555.3195563

1 INTRODUCTION

Many technologists consider chatbots one of the hottest technologies of today [9]. Facebook's release of its Messenger API in 2016 heavily contributed to this expectation, with Facebook reporting already in April 2017 more than 100,000 monthly active bots[1] on the Messenger platform[2]. But also for Twitter in March 2017 Varol et al. [12] estimated that between 9% and 15% of active accounts were bots (29–49 million accounts out of 328 millions[3]). But bots are not confined to Facebook and Twitter; today we can find bots intervening in all kinds of online conversations, from social media, to chats, to Q&A sites, where yesterday there were only humans.

Starting from Weizenbaum's Eliza [13], a simple chatbot developed for psychological conversations, many advanced chatbots have been developed. Notable examples are the Duolingo bot[4] that provides conversational access to language learning resources, Cleverbot[5], an AI-powered bot for generic conversations, and Mitsuku[6], the three-times winner of the Loebner Prize awarded to the winner of a Turing test competition. Going beyond these examples, bots are not limited to generic chats only: Gartner estimates that by 2020 85% of customer requests in companies will be handled by bots [10]. That is, bots are expected to irreversibly permeate both our private and our professional interactions of tomorrow.

While bots are thus expected to – and partly already do – cover a wide spectrum of use cases, what is still missing are what we call *personal chatbots*, i.e., chatbots that are not meant to cater to a general audience but to one user only, chatbots that serve as personal assistants to users, that have intimate knowledge about their preferences, needs and habits, and that are able to cater to them with the necessary pieces of information and assistance. Conceptually, the idea is similar to that of mashups [7], i.e., applications that are developed rapidly by reusing existing functionalities to provide a set of value-adding features to a potentially small user group (even one user only). As for mashups, the high personalization envisioned asks for some form of end-user development [11], which in the case of chatbots benefits from the simplicity of the user interface to be developed: the natural language conversation. The idea is also similar to that of personal assistants like Amazon's Alexa, Apple's Siri and Google's Assistant, which however focus on generic, one-size-fits-all services delivered via voice that are not customizable by non-programmers.

With this position paper, we aim to define what personal chatbots are, what they might be able to do, and how they could be developed. Specifically, in Section 2 we elaborate better on the idea of personal chatbots, providing an intuitive example conversation,

[1] For the sake of this paper, we use the terms "chatbot" and "bot" interchangeably.
[2] https://messenger.fb.com

SE4COG'18, May 28–29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5740-1/18/05…$15.00
https://doi.org/10.1145/3195555.3195563

[3] https://www.recode.net/2017/7/27/16049084/twitter-jack-dorsey-q2-earnings-2017
[4] http://bots.duolingo.com/
[5] http://www.cleverbot.com/
[6] http://www.mitsuku.com/

and we identify a set of requirements that a chatbot needs to meet to be classified as "personal." We use these requirement to discuss options for personal chatbot development and the conception of a suitable end-user development paradigm for personal chatbots (Section 3). Then, we analyze the state of the art, with a special focus on the support that is already available for the development of chatbots (Section 4). We conclude the analysis with an outlook and an outline of our future works.

2 PERSONAL CHATBOTS

2.1 Concept

Let's exemplify what we mean by "personal bot" with the help from a simple yet expressive, fictitious conversation between the user Alice and her personal bot James:

James	**Alice**
	Hi James
Good morning, Alice!	
	Any train delays today?
Let me check... Looks like your train is on time.	
	And the next train?
The next one is 10 minutes late. So, hurry up.	
	Have my eBay orders been shipped?
You have two orders. One has been shipped and should be arriving tomorrow. The other one is still awaiting shipment.	
	Good. Then can you get me the cartoon of the day, please?
Here it is. Enjoy!	

http://bit.ly/2mBJMhL

The conversation could go on like this with or without interruptions, i.e., across multiple usage sessions. What is remarkable in the conversation (see the underlined text) is that – differently from conventional chatbots – James knows very well *who* he is talking to (Alice). Alice can make questions that require prior knowledge about her *train connections* or *eBay orders*, and James knows how to answer and where to retrieve the necessary data from. Retrieving the *cartoon of the day* does not require any particular knowledge about Alice, yet James must know which cartoon she means.

James is truly a personal chatbot, not only because he has all the necessary information about Alice, but also because he has been developed to serve *only* Alice. Alice can "teach" James new features, and she does not have to worry about James telling her personal details (e.g., birthdate or passwords) or even intimate secrets (e.g., who she is watching on Facebook) to anybody else. Some of the services that James provides may be so unique that, among all chatbots that exist, only James is able to provide them (e.g., retrieving the cartoon of the day from Alice's preferred newspaper). By teaching James what she is really interested in, Alice step by step turns James into a personal assistant that is able to answer her very individual needs, for work or private life.

2.2 Requirements

If we carefully analyze the example conversation, we can identify a set of requirements that must be met in order to make James real. First, there are requirements related to the capabilities of the *bot*:

(1) *User identification and authentication:* First and foremost, it is important that the chatbot be able to guarantee that it is interacting with the person it is meant for. That is, the bot must be able to identify users and authenticate their identity to be able to provide personal but also confidential services. James must be sure he is talking to Alice.
(2) *Persistent user profiles* (long-term memory): In order for the bot to distinguish users (if it supports more than one) and to provide custom services, it is necessary to maintain suitable user profiles persistently. The user profiles must not contain only basic user identification data (e.g., Alice's name and birth date), but also take care of user preferences (e.g., possible automated reminders) and long-term conversational context data (e.g., James recalls Alice's preferred newspaper).
(3) *Volatile conversation context* (short-term memory): In order to keep a conversation alive and natural, it is of course necessary that the bot be able to memorize conversational context. For instance, when Alice asks James about the timeliness of the "next train" he must remember that the current conversation is about Alice's train connections.
(4) *Generalization*: Also, for a natural conversation it is important that the bot be able to accept (understand) different utterances or forms of similar questions referencing the same intent, i.e., the purpose of her question. This is where James' cognitive capabilities are challenged and AI comes into play.

Then, there are some requirements related to the *development* of personal bots (*what* Alice should be able to develop):

(5) *Custom questions/answers:* In order to teach the bot new features, it is necessary that the user be able to define the questions (inputs) she would like to have answered and associate possible answers (rendering of outputs) she would like to receive. Alice defined the question for the cartoon of the day and decided how James should answer. Ideally, Alice is asked to provide only one example question, and James is then able to understand also questions that are similar to this example (this again asks for AI).
(6) *Custom intent-action mapping:* In order for custom question-answer pairs to work, i.e., to bind them to some business logic able to compute outputs to visualize, it is necessary that the user be able to associate questions with intents (the goals of the questions) and to map suitable actions to these intents. When James is inquired about trains, he knows that he is asked to retrieve delay data about specific trains.

(7) *Custom actions*: The actions mapped to intents must be able to compute some output in response to inputs collected through the questions. Being these custom, the user must be able to specify own actions to be used to feed answers with meaningful outputs. When interrogated about the trains, James knows where to retrieve the necessary information (e.g., which train website) and how to fetch it (which exact piece of information to extract from the website) and assemble it into an answer (which parameter to fill with the extracted information). Actions may range from simple information retrieval tasks to the intermediation of tasks with effects on the real world, e.g., sending a money transfer via the users' internet banking account or manipulating actuators of the user's smart home.
(8) *Timed actions*: Actions may not necessarily be performed immediately on request. They may as well be enacted at given times (periodically or on fixed dates/times). This is not showcased in the example, but Alice could for instance ask James to tell her each morning before leaving her house whether she should carry an umbrella with her or not, binding the periodic event to a condition to be evaluated before execution. This requires persistent application state.
(9) *Incremental development:* Personal bots are not meant to be a one-shot development effort. Instead, as they "accompany" their user throughout time, they may be developed incrementally as new needs arise. Before using eBay, Alice may not have needed James to check her order statuses; once started using it, she instructed James how to serve her best.

Some other requirements refer to the need for *assisted development* (*how* Alice should be able to develop), given that she is not a programmer with software engineering skills:

(10) *Guided learning*: In order to provide the user with effective support for the identification of new bot capabilities, the development method should proactively suggest the user possible actions, also taking into account emerging patterns of intent-action mappings learned from past development sessions from the same user or from other users.
(11) *Automatic discovery*: The development method should be able to proactively expand the actions provided by the user for a more complete experience. For example, if Alice specifies that she wants to obtain information about trains to a specific location through site x, the method could propose her to integrate the answer with other contextual information, such as weather or temperature information in the selected location.

Of course, next to these requirements turning a bot into a personal chatbot, the development of any chatbot requires the availability of a suitable *communication channel*, the ability to *interpret natural language*, to *identify intents*, to *render answers*, to *manage usage sessions*, etc. For these generic chatbot requirements we redirect the reader to the state of the art.

3 APPROACH

Analyzing the previous requirements, it is important to note that personal chatbots are less a technical problem and more a *methodological* problem. In the state of the art discussion, we will see that for bot development lots of frameworks and APIs exist that help the *developer* to implement a variety of features, from identification and authentication to the management of conversational context. The challenge of personal chatbots rather lies in enabling the *users* of the bots to develop own, custom services to be delivered through the bot. The challenge is thus enabling the users to customize the questions, answers and underlying business logic of their bot – all aspects that do not have ready solutions to rely on. As anticipated, personal chatbots are an *end-user development* problem [11].

3.1 Development Paradigm

The core decision to be taken that affects most how to approach the development of personal chatbots is therefore the decision of how to enable the end-users to develop. That is, the first decision is choosing a suitable development paradigm for end-users. Ideally, the paradigm should enable the end-users to satisfy all the requirements identified in the previous section (plus the generic requirements of bot development), which is not trivial in general. Candidate paradigms are:

- *Coding in standard programming language* (e.g., using JavaScript, Python or similar). State-of-the-art programming languages provide the most flexibility and expressive power and, thus, allow the development of arbitrarily complex bot logics. This is how a professional developer would proceed, but it is also the paradigm that is least suited to end-users who do not have software engineering skills.
- *Coding in domain-specific language* (e.g., using a purposefully developed textual language). We could imagine to devise a programming language that is specifically tailored to the problem of developing bots, e.g., featuring instructions for the specification of questions, answers, actions, intents, action-intent bindings, timed actions, etc. (the language could, for instance, extend AIML, the Artificial Intelligent MarkUp Language[7]). The limited set of instructions would, on the one hand, lower the complexity of the development task but, on the other hand, it would still require the user to learn how to program, a task that only few users would be willing to accomplish.
- *Configuration* (e.g., like in ifttt[8] that provides the user with a set of adapters to connect to most prominent applications and devices). It could be possible to pre-define a set of bot features (e.g., the ability to fetch train status information) and to provide them as pre-built blocks to the users who would only be required to select the blocks of interest and to configure them according to their own needs. However, it would be impossible to pre-can all blocks that the idea of personal chatbots envisions (e.g., it would already be nearly impossible to support all types of transportation means one can imagine) and, hence, the selection of blocks may turn out too restricted to be of any use to a wide group of users.
- *By example* (e.g., by allowing the user to freely navigate the Web and to point at information she is interested in). This idea starts from the assumption that all the information the personal bot should be able to deliver is accessible over the

[7] https://goo.gl/x83UUX
[8] https://ifttt.com/

Web and that the user herself is able to show how to find it in a finite set of navigation steps. In order for the user to add a new functionality to her bot, it would thus suffice to specify the set of triggering questions, the intent behind them, a template for the answer, and to show by example how to obtain the missing information to fill the template, given a possible input to be extracted from the questions. Everything could be carried out live inside the user's browser with the bot being updated on the fly.

In all these scenarios, the user could (i) perform all development tasks *alone* in complete autonomy (perhaps with the help from some development documentation and tutorials), (ii) be assisted by an automated *wizard* with predefined development steps (e.g., one flow for adding a new question, one for defining a new actions, etc.), or – why not? – (iii) be assisted by a dedicated *chatbot* walking the user through the different development steps in response to the user's input and requests.

Considering the expected ease of development, in this paper we advocate for a *bot-assisted, by-example paradigm* that does not require any particular software development skills, does not leave the user alone (and possibly frustrated) in her task, and still has the power of supporting a huge variety of functionalities (everything that is Web-accessible can be used to create conversations).

3.2 Architecture

Recalling the requirements identified above, it seems reasonable to start from a multi-tenant *platform* for personal chatbot development able to cater chatbots to multiple users at the same time upon configuration. Most importantly, a dedicated platform enables separating generic infrastructure services from bot-specific configurations. The former can be provided once for all, the latter can be provided by the user via a purposefully developed client-side development environment.

Figure 1 illustrates a possible functional architecture. The *user manager* provides for basic user management (R2) and access control (R1). The *personal bot engine* comes with built-in support for context management in conversations (R3) and orchestrates the *input parser*, the enactment of suitable *actions* (possibly making use of a *headless browser* to mimic recorded user interactions), and the *output synthesizer*. The input parser is responsible for providing support for generalization in the discourse (R4), e.g., by leveraging on existing NLP libraries or suitable cognitive APIs. Examples of basic actions supported are filling a form field, clicking a button, extracting a piece of information, and similar. The *new feature crawler* proactively looks for new bot capabilities that can be suggested (recommended) to the user at design and runtime (R11), for example based on similarity measures for intents. All these components may internally leverage on state-of-the-art machine learning (ML) and artificial intelligence (AI) support able to provide, for instance, for advanced language parsing and interpretation.

The design of a new personal chatbot could happen directly inside the user's browser with the help from a dedicated *bot development browser extension* able to collect questions, intents and answer templates (R5) from the user, e.g., via simple HTML forms and natural language conversation. Once questions and intents are collected, the user is able to manage the respective relationships and

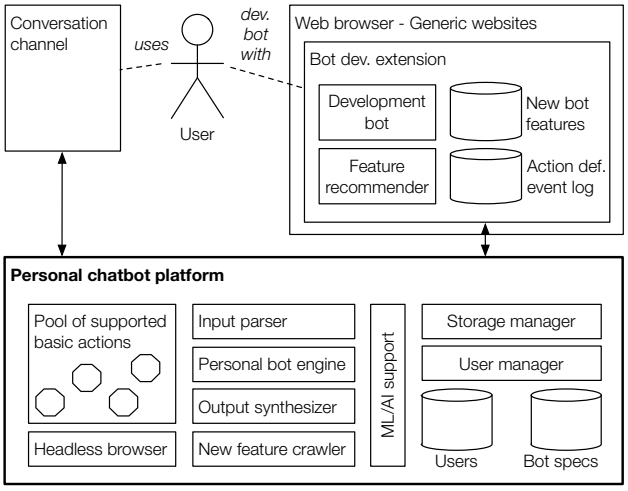

Figure 1: Functional architecture underlying personal chatbots: runtime platform and development environment.

to declare and associate actions to intents (R6). New features are defined by recording exemplary user navigations and actions directly inside the browser, allowing the user to parametrize the recordings, to assign default values, to identify outputs, and to map them to intents and answer templates directly inside the browser extension (R7). Newly defined features are transparently sent from the browser extension to the platform and integrated into the user's existing *bot specifications* by a dedicated *storage manager*. A bot specification contains example questions, intents, recorded/specified actions, intent action-mappings, answer templates. The *action definition event log* captures user interaction events about how the user specifies new actions, in order to provide support for guided learning (R10), e.g., using a guided or wizard-like paradigm also able to recognize if specific corner cases should be considered. The *feature recommender* pushes recommendations by the new feature crawler to the user (R11).

The extension and bot engine also support actions that are not directly mapped to any intent at design time, but that can be associated with time-dependent invocation logics at runtime. The necessary vocabulary to define, for example, periodical events (e.g., "Please, tell me the weather every morning") is a built-in feature of the supporting platform and does not require any additional specification from the user (R8). The extension further provides for *bot capability management functions*, such as adding, deleting, modifying questions, answers, intents and actions, effectively enabling a step-by-step specification of the bot features (R9).

4 STATE OF THE ART

Although the current interest in chatbots is still rather young, there is already a wide spectrum of platforms, libraries, services and APIs that significantly ease the development of chatbots.

In terms of *dedicated platforms*, several of them support the rapid development of chatbots. For example, Dialogflow[9] (formerly api.ai)

[9] https://dialogflow.com/

and Motion AI[10]) offer a visual, flowchart-based development paradigm for the definition of questions, intent interpretations, actionable answers and their assembly into contextual conversation flows. Other platforms use a proprietary, textual language for conversation design. For example, PandoraBots[11] proposes AIML (Artificial Intelligent MarkUp Language), an XML-based language for tagging conversation elements, such as parameters in the user input, query patterns and answer templates. Microsoft's Bot Framework[12] rather bets on conventional programming languages using the .NET family of languages. IBM's Watson Conversation Service follows a similar approach and comes with SDKs for Node, Java, Python, .NET. Most of these frameworks are already equipped with advanced AI and NLP support that only needs to be properly configured.

Also, most chatbot platforms enable the development of *multi-channel chatbots*, i.e., chatbots that can be accessed through different conversational channels, such as Facebook Messenger, WhatsApp, SMS, web chats, social media, but also voice and text. Others, for example FlowXo[13], provide for different deployments each one specific for a given channel.

Some platforms offer *open architectures*, enabling the invocation of external APIs for answering user queries; this enhances, for example, their integrability into proprietary information systems. For example, with DialogFlow one can specify that some parameters of the user queries can be used to invoke external services and thus to retrieve content from outside the platform. The exchange of data (input parameters, result sets) in Json is also possible.

The capability to *interpret natural language* is also very different. The platforms offered by the big providers (e.g., Microsoft and IBM) offer the possibility to connect to natural-language processing engines. In other frameworks, the interpretation of the user queries and the identification of intents strongly depend on the capability of the developer to adequately define at design time the different dialogue elements and the dialogue flow. Yet, the developer has not to start from scratch and invent own NLP software. There is a myriad of NLP APIs or libraries that allow everyone to use powerful language processing capabilities remotely. wit.ai[14] is an example of remote API, rasa NLU[15] is an example of locally installable library.

Considering the idea of *personal chatbots*, it is evident that the state of the art in bot technology nicely covers the generic bot development requirements (basic conversation management, different channels, context management, etc.) and also requirements R1–R4. That is, significant work can be reused for the development of personal chatbots. What is still missing is a development paradigm oriented toward end-users, both from a customization point of view and from a development paradigm point of view. Requirements R5–R11 derived in Section 2.2 become central. In this respect, it may be possible to capitalize on results achieved in the area of end-user development, more specifically in the field of mashups [4, 5, 8]. For instance, Aghaee and Pautasso [1, 2] already studied the use of natural language for specifying information needs to be fulfilled by a mashup under composition; however, the approach was based on a restricted, structured subset of natural language only.

As for the *personal assistants* Alexa, Siri and Assistant proposed by Amazon, Apple and Google, respectively, it is important to note that they too could be used as delivery channel of the personalized services proposed in this paper, since they all provide suitable extension mechanisms and SDKs. For instance, it could be possible to imagine Alice talking to James via Google's Assistant by invoking James using the specific command "Ok Google, talk to James," starting from which Assistant would leverage on the capabilities of James to deliver personalized services. That is, while the idea of this paper is proposed as a complement of these personal assistants, it could as well be integrated into them as an extension. Of course, as of now, what is missing in all these three examples of commercial personal assistants is the possibility for end-users to personalize their capabilities beyond the possibility to enact pre-canned applications developed by programmers.

5 OUTLOOK AND CHALLENGES

With this paper, we launch the idea of personal bots, so personal that they have to be "programmed" by the users themselves. We have discussed how the intuitiveness of the natural language paradigm can help end-users *i)* explore the capabilities of a chatbot, *ii)* develop bots that really respond to their needs, and *iii)* satisfy these needs on the go. It is important to note that the natural language paradigm eases end-user development, as it does not require users to be able to design graphical UIs or data presentation formats (like in most mashups), a task that can easily get complex. On the other hand, using a conversational paradigm also for development, user intents may not be trivial and evolve along the same or different interaction sessions, both during development and during use. Thanks to cutting-edge cognitive applications now available it is possible to understand intents with confidence and assist the users also in evolving contexts. How to best leverage on these capabilities is a challenge to be addressed in our future work.

Another big challenge is understanding how the conversational paradigm can integrate or even replace the traditional, visual paradigm when it comes to data exploration. Widget-based, visual user interfaces have proven capable of lowering the entry barriers to users who want to access heterogeneous data in an integrated fashion [3, 6]. Effectively presenting huge amounts of data to users in a conversational interface is instead a significant challenge. Providing them with an integrated view over different, related data sources is even more complex. This is not only a problem of data presentation, but also of adequate data analysis techniques, such as summarization techniques, interactive drill-down and roll-up data exploration capabilities or the automatic generation of graphical charts. The natural language paradigm is powerful as long as questions and answers are simple. If an answer wants to provide insight into large datasets, effective bot-compliant aggregation techniques are needed. Our future work will study how these needs impact the different layers (data, application logic, presentation) in the development and interaction with personal bots.

Acknowledgement. We would like to thank Stefano Sanitate and Paolo Roncaglioni for their input and the fruitful discussions.

[10] https://www.motion.ai/
[11] https://www.pandorabots.com/
[12] https://docs.botframework.com/en-us/
[13] https://flowxo.com/
[14] https://wit.ai/
[15] https://nlu.rasa.ai/

REFERENCES

[1] Saeed Aghaee and Cesare Pautasso. 2014. End-User Development of Mashups with NaturalMash. *J. Vis. Lang. Comput.* 25, 4 (2014), 414–432. https://doi.org/10.1016/j.jvlc.2013.12.004

[2] Saeed Aghaee, Cesare Pautasso, and Antonella De Angeli. 2013. Natural End-User Development of Web Mashups. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*. IEEE, 111–118.

[3] Cinzia Cappiello, Maristella Matera, and Matteo Picozzi. 2015. A UI-Centric Approach for the End-User Development of Multi-device Mashups. *TWEB* 9, 3 (2015), 11. https://doi.org/10.1145/2735632

[4] Cinzia Cappiello, Maristella Matera, Matteo Picozzi, Gabriele Sprega, Donato Barbagallo, and Chiara Francalanci. 2011. DashMash: A Mashup Environment for End User Development. In *Proc. of ICWE 2011 (LNCS)*, Vol. 6757. Springer, 152–166.

[5] Florian Daniel. 2015. Live, Personal Data Integration Through UI-Oriented Computing. In *Proc. of ICWE 2015*. 479–497. https://doi.org/10.1007/978-3-319-19890-3_31

[6] Florian Daniel, Fabio Casati, Boualem Benatallah, and Ming-Chien Shan. 2009. Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. In *Conceptual Modeling - ER 2009 (LNCS)*, Vol. 5829. Springer, 428–443.

[7] Florian Daniel and Maristella Matera. 2014. *Mashups - Concepts, Models and Architectures*. Springer. https://doi.org/10.1007/978-3-642-55049-2

[8] Florian Daniel, Maristella Matera, and Michael Weiss. 2011. Next in Mashup Development: User-Created Apps on the Web. *IT Professional* 13, 5 (2011), 22–29.

[9] Matt Grech. 2017. The Current State of Chatbots in 2017. *GetVoIP.com* (April 2017). https://getvoip.com/blog/2017/04/21/the-current-state-of-chatbots-in-2017/

[10] Inbenta Technologies Inc. 2016. *The Ultimate Guide to Chatbots for Businesses*. Technical Report. www.inbenta.com.

[11] Henry Lieberman, Fabio PaternÚ, and Volker Wulf. 2004. *End User Development*. Human-Computer Interaction Series, Vol. 9. Springer.

[12] Onur Varol, Emilio Ferrara, Clayton A Davis, Filippo Menczer, and Alessandro Flammini. 2017. Online human-bot interactions: Detection, estimation, and characterization. *arXiv preprint arXiv:1703.03107* (2017).

[13] Joseph Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM* 9, 1 (1966), 36–45.